UNIVERSITÉ Clermont Auvergne

SD Aurillac Science des données Cybersécurité

RAPPORT DE PROJET
Architecture d'Intégration de Données

Mise en place d'un Data Warehouse

Gestion des Ventes et Logistiques

Auteurs :
Adebola Fonton
Otiniel Aguida
Hugues Fangnon

Professeur référent :
M. Bruno Lagarde

Année Universitaire 2025-2026

Table des matières

Introduction

Dans le cadre de ce projet, une architecture d'intégration de données a été mise en œuvre pour une entreprise de vente en ligne. L'objectif principal était de centraliser les données relatives aux ventes et aux livraisons, issues des bases de données opérationnelles de l'entreprise, au sein d'un entrepôt de données (Data Warehouse) localisé au siège.

Cet entrepôt de données constitue un outil stratégique permettant de regrouper, analyser et restituer les informations essentielles à la gestion et au pilotage des activités commerciales et logistiques.

Objectif du Projet

L'objectif principal de ce projet est de concevoir et de mettre en œuvre un entrepôt de données permettant de centraliser les informations relatives aux ventes et aux livraisons, afin de faciliter les analyses destinées aux services Finance et Logistique. Il vise également à automatiser le processus d'extraction, de transformation et de chargement (ETL) des données depuis la base opérationnelle située à Aurillac vers le Data Warehouse implanté à Paris. Enfin, ce projet a pour ambition de proposer une architecture évolutive et flexible, capable de soutenir l'analyse des performances commerciales et logistiques de l'entreprise.

Contexte Choisi

Il a été supposé que la base de données opérationnelle est située dans la ville d'Aurillac, tandis que l'entrepôt de données, localisé au siège de l'entreprise, se trouve à Paris. La mise en place du processus d'extraction, de transformation et de chargement (ETL) des données d'Aurillac vers le Data Warehouse de Paris démontre la faisabilité d'une telle architecture et permettrait d'étendre ce mécanisme à des données provenant d'autres sites géographiques.

Les outils retenus pour la réalisation de ce projet sont **PostgreSQL**, utilisé pour la création et la gestion des bases de données source et cible, ainsi que **Talend**, employé comme solution ETL.

Chapitre 1

Analyse du Système d'Information Opérationnel (SIO)

1.1 Analyse des données d'Aurillac

La base de données opérationnelle située à Aurillac regroupe l'ensemble des informations nécessaires au fonctionnement quotidien de l'entreprise. Elle comprend notamment les tables suivantes :

- **Catégorie :** informations relatives aux catégories de produits.
- **Client :** données descriptives des clients (nom, adresse, coordonnées).
- **Commande :** détails des commandes, dates et transporteur.
- **DétailCommande :** produits commandés (quantités, prix unitaires, remises).
- **Produit :** caractéristiques, référence, nom, fournisseur.
- **Employé :** informations relatives aux employés.
- **Fournisseur :** données concernant les entreprises d'approvisionnement.
- **Messager :** sociétés chargées de la livraison.

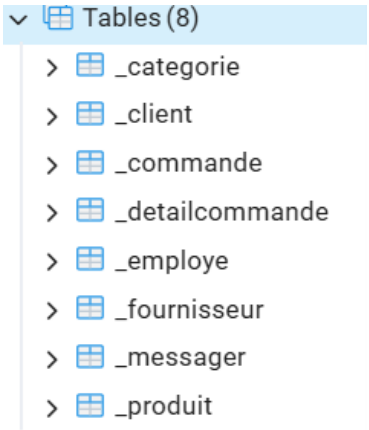


FIGURE 1.1 – Aperçu des données ou tables d'Aurillac

1.2 Modèle Conceptuel de Données (MCD)

Le modèle conceptuel de donnée de cette base se présente comme suit :

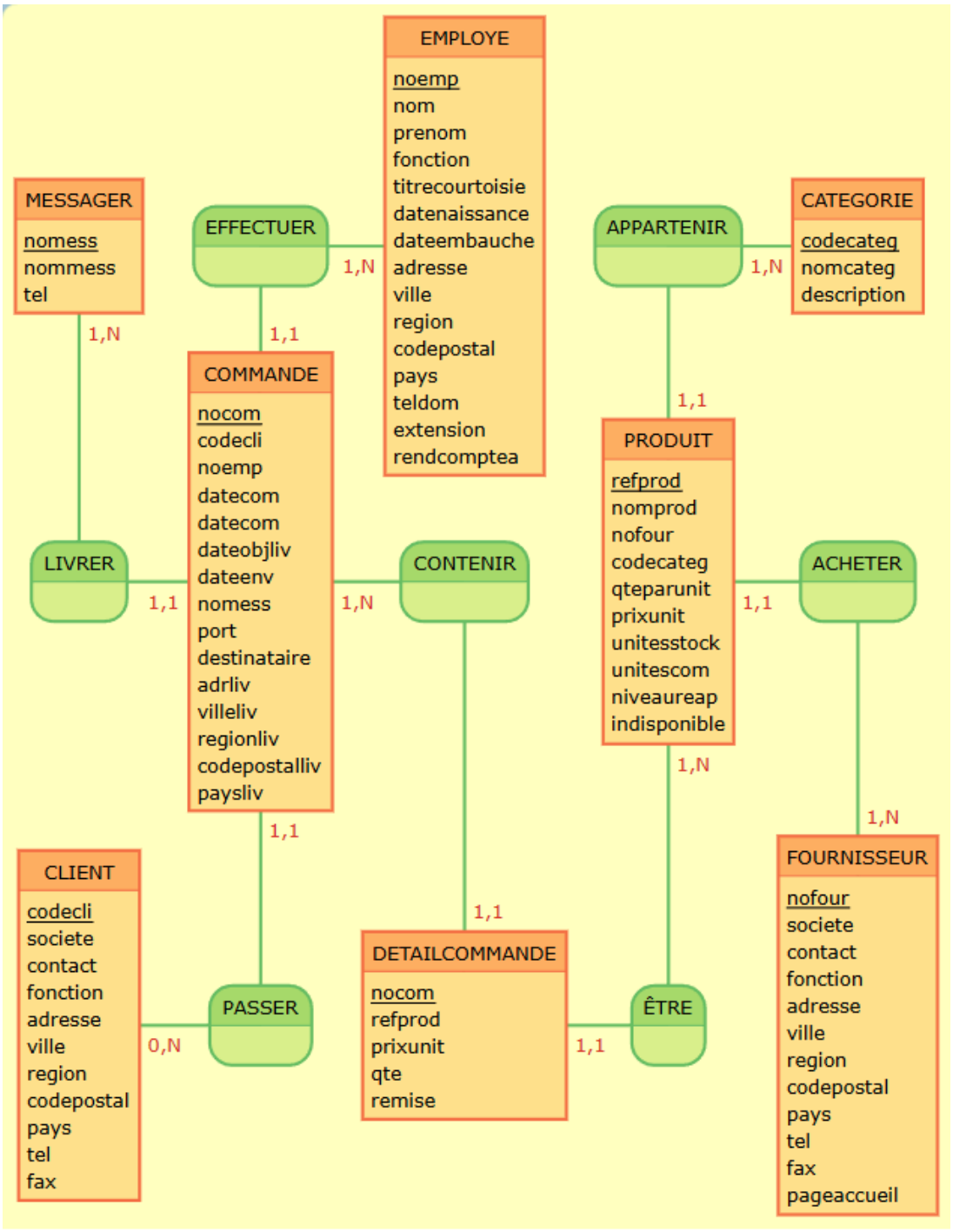


FIGURE 1.2 – MCD des 8 tables du SIO

Chapitre 2

Réponse aux demandes de gestion

2.1 Gestion des Ventes (Service Finances)

Afin d'aider le service finances, nous avons choisi de faire ressortir les données suivantes :

- **Chiffre d'affaires par client :** Pour identifier les clients stratégiques et personnaliser les offres.
- **Chiffre d'affaires par pays :** Pour décider de l'extension ou du retrait d'un marché.
- **Chiffre d'affaires par ville :** Vision fine pour le marketing local.
- **Chiffre d'affaires par employé :** Pour évaluer la performance commerciale.
- **Chiffre d'affaires par catégorie :** Pour ajuster l'offre produit.

2.2 Gestion des Livraisons (Service Logistique)

Pour aider le service logistique, nous avons choisi de faire ressortir :

- **Délai de livraison :** Pour évaluer la qualité globale du service.
- **Délai moyen de livraison par messager :** Pour comparer les performances des transporteurs.
- **Nombre de livraisons par pays :** Pour optimiser les coûts de transport internationaux.
- **Produits les plus livrés :** Pour anticiper les volumes et les stocks.

Chapitre 3

Conception du Système d'Information Décisionnel

3.1 Schéma en Étoile : Gestion des Ventes

Pour répondre aux besoins du service Finances, nous avons modélisé un premier datamart dédié à l'analyse des ventes.

Ce modèle s'articule autour de la table de faits **_faits_gestion_ventes**, qui centralise les métriques de performance telles que le prix unitaire, la quantité vendue et la remise accordée. Elle contient également les clés étrangères permettant de faire le lien avec les axes d'analyse.

Les dimensions choisies offrent plusieurs perspectives d'analyse :

— **Client :** pour analyser qui achète (société, localisation).
— **Produit et Catégorie :** pour analyser ce qui est vendu.
— **Commande :** pour analyser le contexte de la vente (dates, lieux de livraison).
— **Employé :** pour analyser la performance des vendeurs.

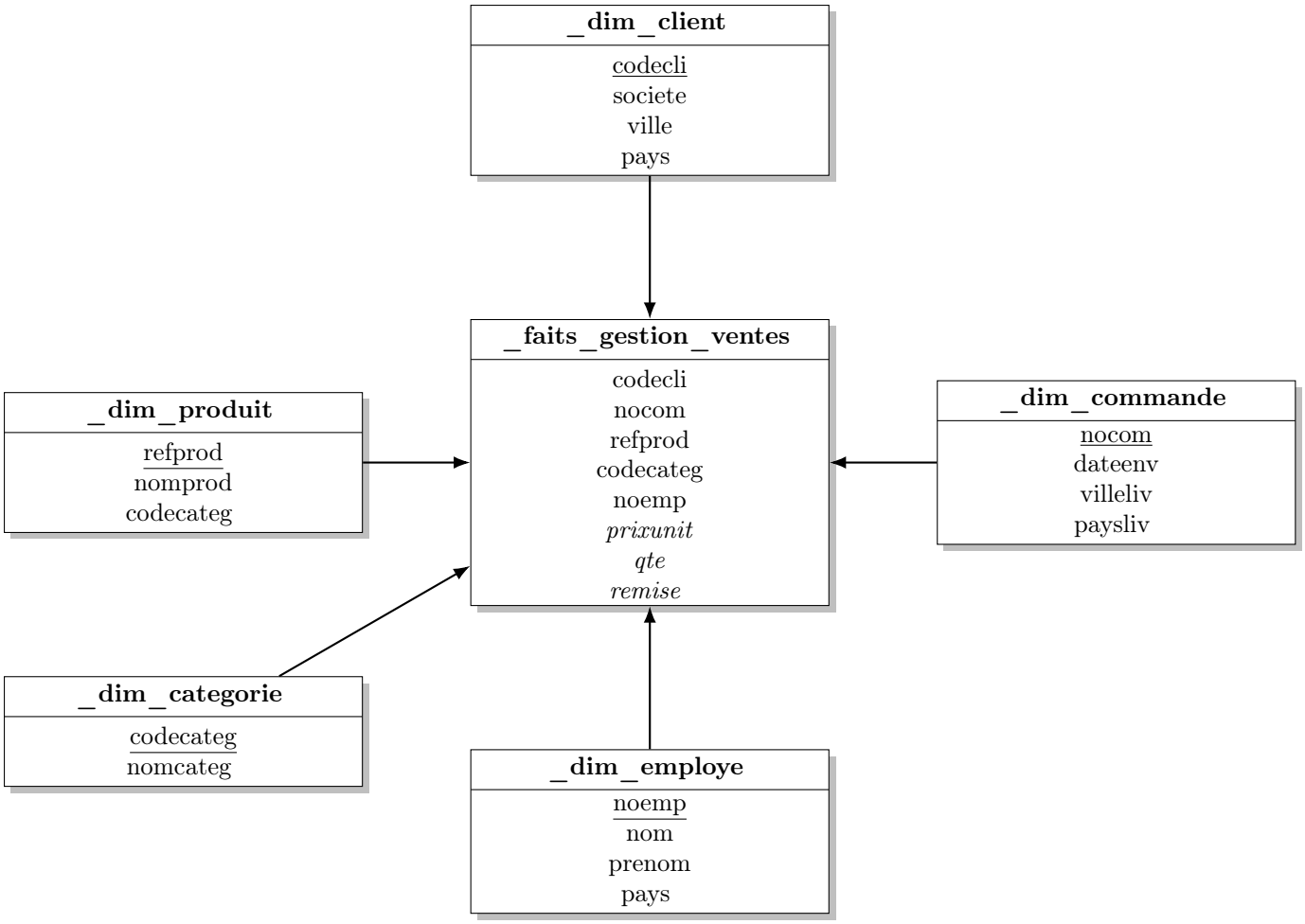


FIGURE 3.1 – Schéma en étoile - Gestion des Ventes

Code SQL de création des tables (Ventes)

```sql
CREATE TABLE public._faits_gestion_ventes (
    codecli character varying,
    nocom integer,
    refprod smallint,
    codecateg smallint,
    noemp smallint,
    prixunit smallint,
    qte smallint,
    remise numeric
);

CREATE TABLE public._dim_produit (
    refprod smallint,
    nomprod character varying,
    codecateg smallint
);

CREATE TABLE public._dim_employe (
    noemp smallint,
    nom character varying,
    prenom character varying,
    pays character varying
```

```
);

CREATE TABLE public._dim_commande (
    nocom integer,
    dateenv character varying,
    villeliv character varying,
    paysliv character varying
);

CREATE TABLE public._dim_client (
    codecli character varying,
    societe character varying,
    ville character varying,
    pays character varying
);

CREATE TABLE public._dim_categorie (
    codecateg smallint,
    nomcateg character varying
);
```

Listing 3.1 – DDL Gestion des Ventes

3.2 Schéma en Étoile : Gestion des Livraisons

Pour répondre aux besoins du service Logistique, nous avons conçu un second modèle centré sur la table de faits `_faits_gestion_livraisons`. Elle permet d'analyser les délais et les flux logistiques grâce aux dimensions Messager, Produit et Commande.

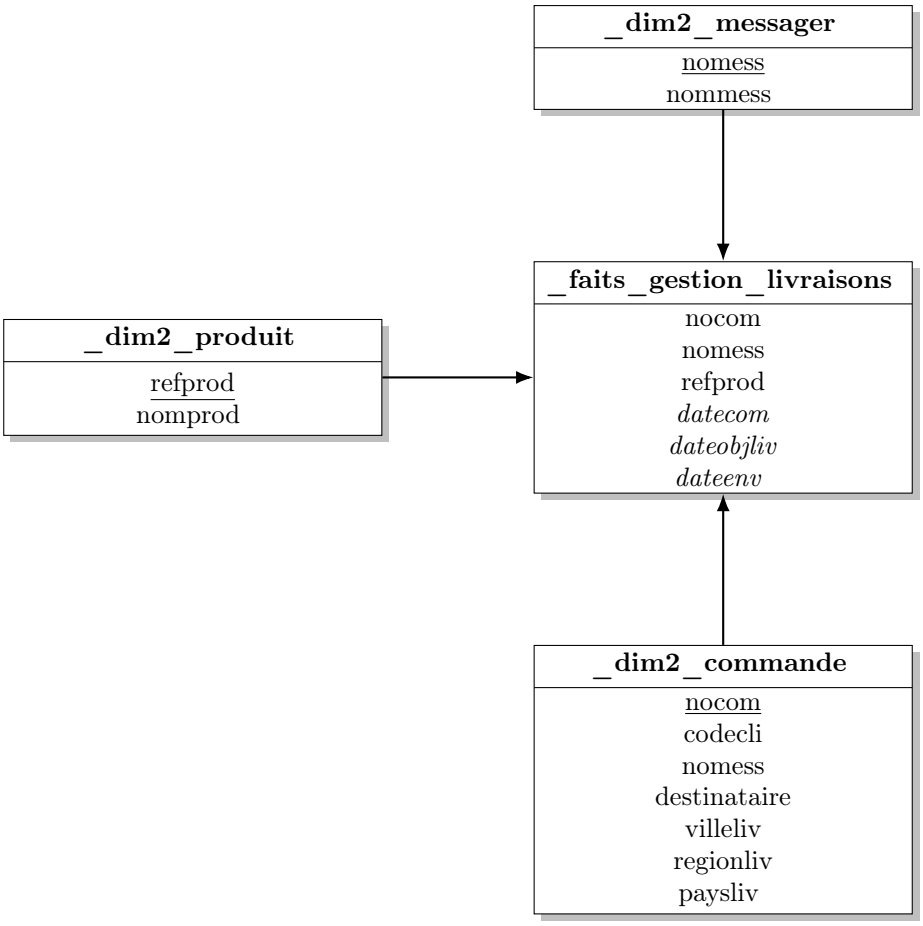


FIGURE 3.2 – Schéma en étoile - Gestion des Livraisons

Code SQL de création des tables (Livraisons)

```sql
CREATE TABLE public._faits_gestion_livraisons (
    nocom integer,
    nomess smallint,
    refprod smallint,
    datecom character varying,
    dateobjliv character varying,
    dateenv character varying
);

CREATE TABLE public._dim2_messager (
    nomess smallint,
    nommess character varying
);

CREATE TABLE public._dim2_commande (
    nocom integer,
    codecli character varying,
    nomess smallint,
    destinataire character varying,
    villeliv character varying,
    regionliv character varying,
    paysliv character varying
);

```

```sql
CREATE TABLE public._dim2_produit (
    refprod smallint,
    nomprod character varying
);
```

Listing 3.2 – DDL Gestion des Livraisons

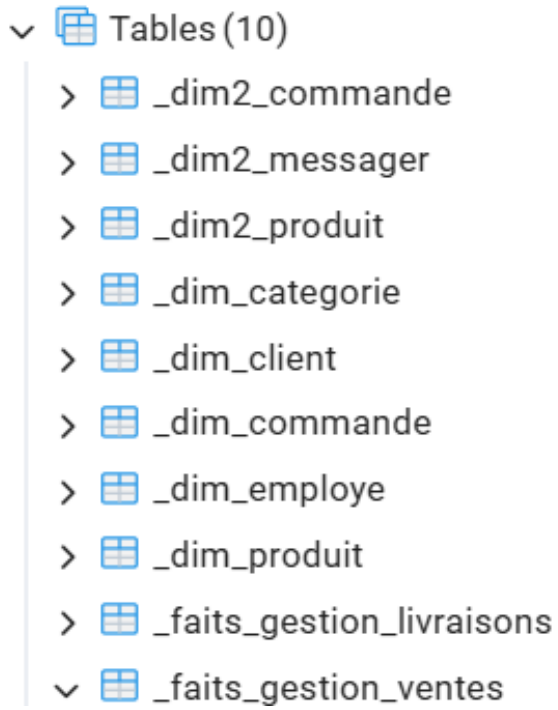


FIGURE 3.3 – Résultat de la création des tables dans PostgreSQL

Chapitre 4

Processus ETL

Le processus ETL (Extract, Transform, Load) constitue la colonne vertébrale de notre système décisionnel. Il assure le transfert et la conversion des données depuis la base opérationnelle d'Aurillac (SIO) vers l'entrepôt de données situé à Paris (SID).

Nous utilisons l'outil **Talend Open Studio** pour orchestrer ces flux. Chaque flux est modélisé sous forme de "Job" qui extrait les données via des requêtes SQL, les transforme si nécessaire, et les charge dans la base cible.

4.1 Flux d'alimentation : Gestion des Ventes

Ce premier job a pour but d'alimenter la table de faits `_faits_gestion_ventes`.

4.1.1 Design du Job Talend

Le job commence par un composant d'extraction (`tDBInput`) qui exécute la requête SQL complexe. Les données passent ensuite par un `tMap` (pour d'éventuels mappages ou conversions de types) avant d'être insérées dans la table cible via un `tDBOutput`.

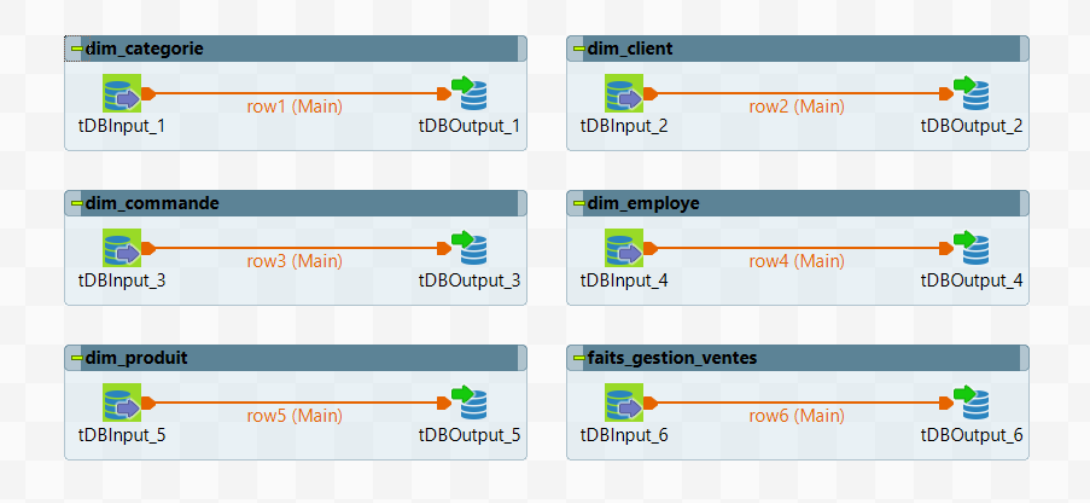


FIGURE 4.1 – Vue graphique du Job Talend pour les Ventes

4.1.2 Requête d'extraction utilisée

La requête SQL injectée dans le composant `tDBInput` est la suivante. Elle réalise la jointure entre les commandes, les clients et les produits pour consolider les données.

```sql
SELECT cli.codecli,
    com.nocom,
    dc.refprod,
    cat.codecateg,
    em.noemp,
    dc.prixunit,
    dc.qte,
    dc.remise
FROM public._client cli, public._commande com, public._detailcommande dc
   ,
    public._categorie cat, public._produit pr, public._employe em
WHERE com.codecli = cli.codecli
AND dc.nocom = com.nocom
AND dc.refprod = pr.refprod
AND cat.codecateg = pr.codecateg
AND em.noemp = com.noemp;
```

Listing 4.1 – Extraction SQL pour la table de faits Ventes

4.2 Flux d'alimentation : Gestion Logistique

Ce second flux se concentre sur les délais de livraison. Il alimente la table `_faits_gestion_livraison`

4.2.1 Design du Job Talend

De la même manière, ce job extrait les données temporelles et logistiques. Il est crucial ici de bien gérer les formats de dates lors du transfert entre la base source et la base cible.

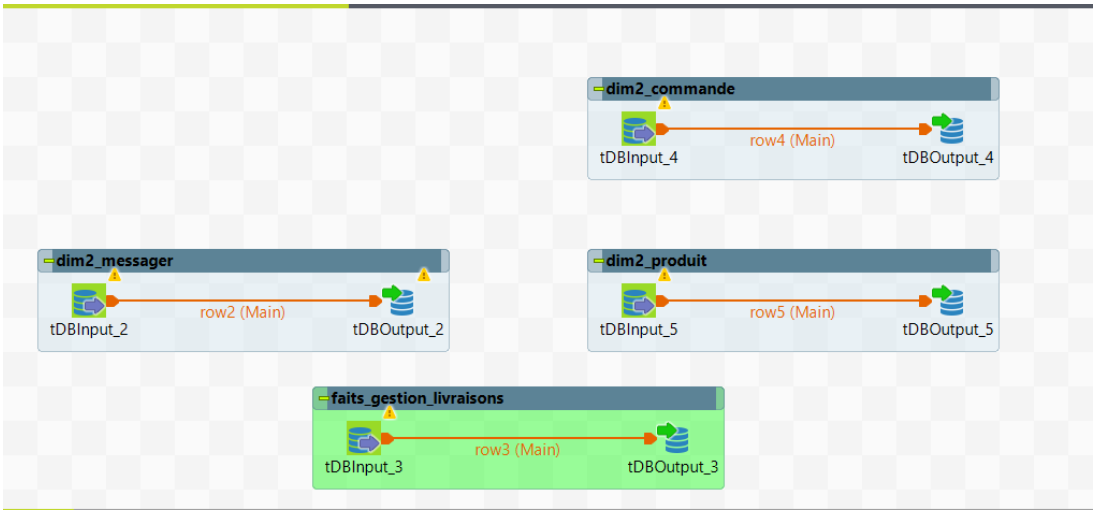


FIGURE 4.2 – Vue graphique du Job Talend pour la Logistique

4.2.2 Requête d'extraction utilisée

Cette requête récupère les dates clés (commande, objectif, envoi) et lie les produits à leurs transporteurs.

```
SELECT com.nocom,
    me.nomess,
    pr.refprod,
    com.datecom,
    com.dateobjliv,
    com.dateenv
FROM public._commande com, public._messager me, public._produit pr,
   public._detailcommande dc
WHERE com.nomess = me.nomess
AND com.nocom = dc.nocom
AND dc.refprod = pr.refprod;
```

Listing 4.2 – Extraction SQL pour la table de faits Livraisons

Chapitre 5

Requêtes de Reporting (Data Marts)

5.1 Indicateurs de Vente

Les vues suivantes ont été créées pour alimenter le tableau de bord Excel du service Finances.

```sql
-- CA par Client
CREATE VIEW public.CAbycli AS
SELECT codecli,
   sum((prixunit * qte)::numeric * (1::numeric - remise)) AS CAclient
FROM _faits_gestion_ventes fg GROUP BY codecli;

-- CA par Pays
CREATE VIEW public.cabypays AS
SELECT dimcli.pays, sum((prixunit * qte) * (1 - remise)) AS capays
FROM _faits_gestion_ventes fg
JOIN _dim_client dimcli ON dimcli.codecli = fg.codecli
GROUP BY dimcli.pays ORDER BY capays DESC;

-- CA par Ville
CREATE VIEW public.cabyville AS
SELECT dimcli.ville, sum((prixunit * qte) * (1 - remise)) AS caville
FROM _faits_gestion_ventes fg
JOIN _dim_client dimcli ON dimcli.codecli = fg.codecli
GROUP BY dimcli.ville ORDER BY caville DESC;

-- CA par Categorie
CREATE VIEW public.cabycategorie AS
SELECT dimcat.codecateg, dimcat.nomcateg,
   sum((prixunit * qte) * (1 - remise)) AS cacategorie
FROM _faits_gestion_ventes fg
JOIN _dim_categorie dimcat ON dimcat.codecateg = fg.codecateg
GROUP BY dimcat.codecateg, dimcat.nomcateg ORDER BY cacategorie DESC;
```

Listing 5.1 – Vues SQL pour les ventes

5.2 Indicateurs Logistiques

Les vues suivantes alimentent le tableau de bord Logistique.

```sql
-- 1. Calcul du d lai de livraison brut
CREATE VIEW delailivraison AS
```

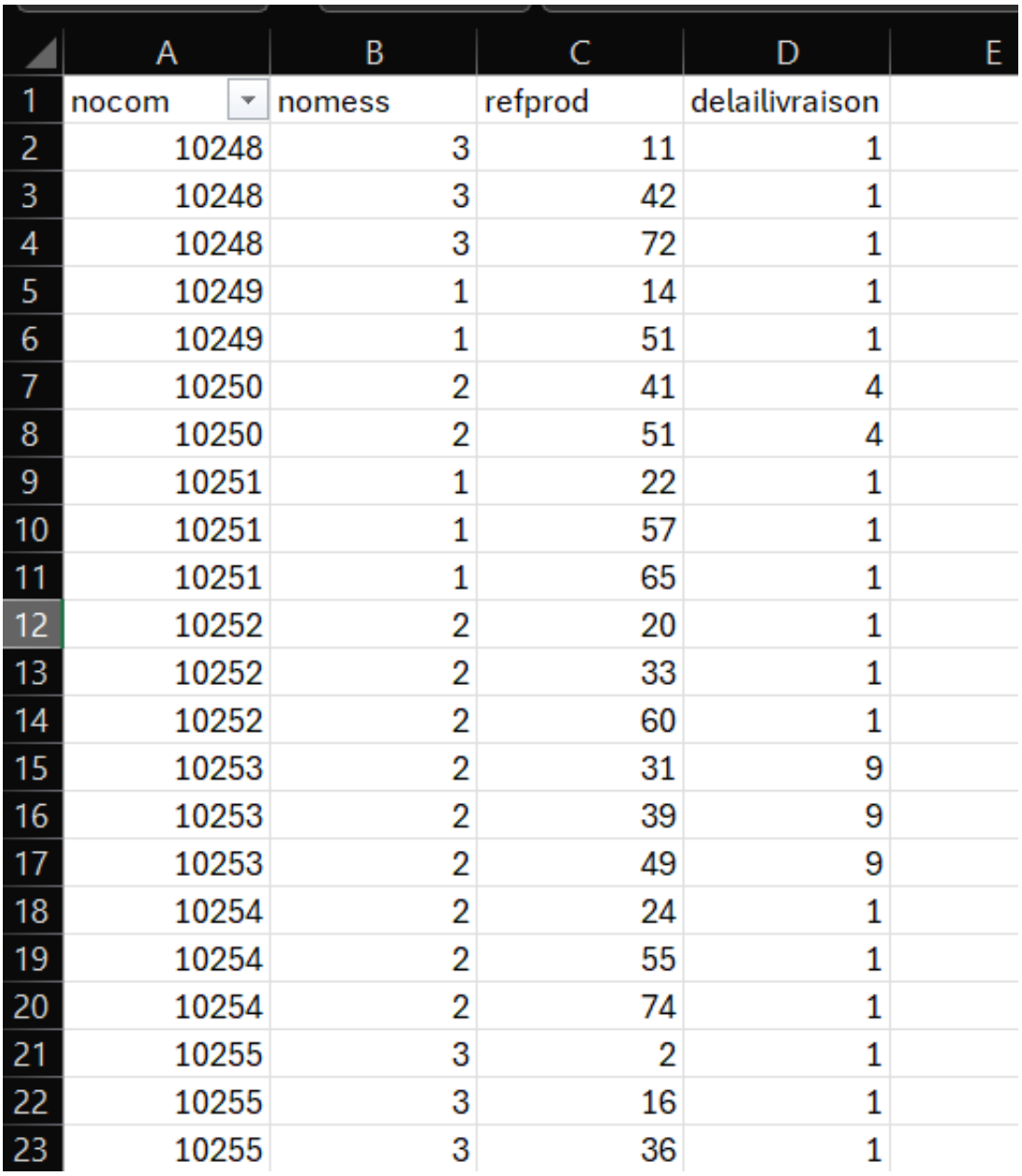
```sql
SELECT nocom,
    nomess,
    refprod,
    (dateenv::date - datecom::date) AS delailivraison
FROM public._faits_gestion_livraisons;

-- 2. D lai moyen de livraison par messager
CREATE VIEW delaimoyenlivraisonbymessager AS
SELECT nomess,
    CEIL(AVG(dateenv::date - datecom::date)) AS
   delaimoyenlivraisonbymessager
FROM public._faits_gestion_livraisons
GROUP BY nomess;

-- 3. Nombre de livraisons par pays
CREATE VIEW nblivraisonparpays AS
SELECT paysliv,
    COUNT(nocom) AS nblivraisonparpays
FROM _dim2_commande
GROUP BY paysliv;

-- 4. Nombre de livraisons par pays et par messager
CREATE VIEW nblivraisonparpaysparmess AS
SELECT dimcom.nomess,
    dimmess.nommess,
    dimcom.paysliv,
    COUNT(dimcom.nocom) AS nblivraisonparpaysparmess
FROM _dim2_commande dimcom
JOIN _dim2_messager dimmess ON dimcom.nomess = dimmess.nomess
GROUP BY dimcom.nomess, dimmess.nommess, dimcom.paysliv;

-- 5. Nombre total de livraisons par messager
CREATE VIEW nblivraisonparmess AS
SELECT dimcom.nomess,
    dimmess.nommess,
    COUNT(dimcom.nocom) AS nblivraisonparmess
FROM _dim2_commande dimcom
JOIN _dim2_messager dimmess ON dimcom.nomess = dimmess.nomess
GROUP BY dimcom.nomess, dimmess.nommess;

-- 6. Les produits les plus livr s
CREATE VIEW produitslespluslivre AS
SELECT fgl.nocom,
    dimpro.nomprod,
    COUNT(dimpro.refprod) AS nblivree
FROM _dim2_produit dimpro
JOIN _faits_gestion_livraisons fgl ON fgl.refprod = dimpro.refprod
GROUP BY fgl.nocom, dimpro.refprod, dimpro.nomprod
ORDER BY nblivree DESC;
```

Listing 5.2 – Vues SQL pour les indicateurs logistiques

5.3 Exemple de Restitution (Excel)

Afin de rendre ces données exploitables par les décideurs, les résultats des vues SQL sont exportés automatiquement vers Excel via Postgres.

L'image ci-dessous illustre le fichier final obtenu pour l'analyse du délai de livraison en jours . Ce format permet au service Logistiques de réaliser des tris, des graphiques ou des tableaux croisés dynamiques sans avoir besoin de compétences techniques en SQL.



	A	B	C	D	E
1	nocom	nomess	refprod	delailivraison	
2	10248	3	11	1	
3	10248	3	42	1	
4	10248	3	72	1	
5	10249	1	14	1	
6	10249	1	51	1	
7	10250	2	41	4	
8	10250	2	51	4	
9	10251	1	22	1	
10	10251	1	57	1	
11	10251	1	65	1	
12	10252	2	20	1	
13	10252	2	33	1	
14	10252	2	60	1	
15	10253	2	31	9	
16	10253	2	39	9	
17	10253	2	49	9	
18	10254	2	24	1	
19	10254	2	55	1	
20	10254	2	74	1	
21	10255	3	2	1	
22	10255	3	16	1	
23	10255	3	36	1	

FIGURE 5.1 – Exemple de Data Mart exporté sous Excel (délai de livraison)

Conclusion

En somme, la réalisation de ce projet d'intégration de données constitue une avancée majeure pour le pilotage stratégique de l'entreprise. Au-delà de la simple centralisation technique entre le site opérationnel d'Aurillac et le siège parisien, nous avons bâti une véritable infrastructure décisionnelle robuste et pérenne.

La mise en œuvre des flux ETL via Talend a permis de structurer et de sécuriser le transfert de l'information. Cette démarche garantit la cohérence des données entre les systèmes et fournit une base saine pour l'analyse. Désormais, les départements Finances et Logistique disposent d'une autonomie d'analyse complète et d'une réactivité inédite face aux indicateurs de performance (KPI).

Ce Data Warehouse ne constitue pas une finalité, mais un socle évolutif : son architecture modulaire est prête à absorber la croissance des volumes et à intégrer de futures sources de données, inscrivant ainsi l'entreprise dans une véritable démarche de Business Intelligence.