Université Djillali Liabes

Sidi Bel Abbès

TP4 Data Warehouse

Binome :

Berhail Sami : berhailsami@outlook.fr

Brahimi Rafik : Brahimi9rafik@gmail.com

Table des matières

1 Introduction :

Une chaîne de pharmacies souhaite mettre en place un entrepôt de données pour améliorer son processus décisionnel. L'entreprise dispose actuellement d'une base de données transactionnelle (OLTP) gérant les opérations quotidiennes de ses différentes pharmacies. L'objectif est de construire un entrepôt de données permettant d'analyser les performances et les tendances sur plusieurs axes.

2 Partie 1 : Conception du model dimensionnel

2.1 Identification des tables des faits et des tables de dimensions :

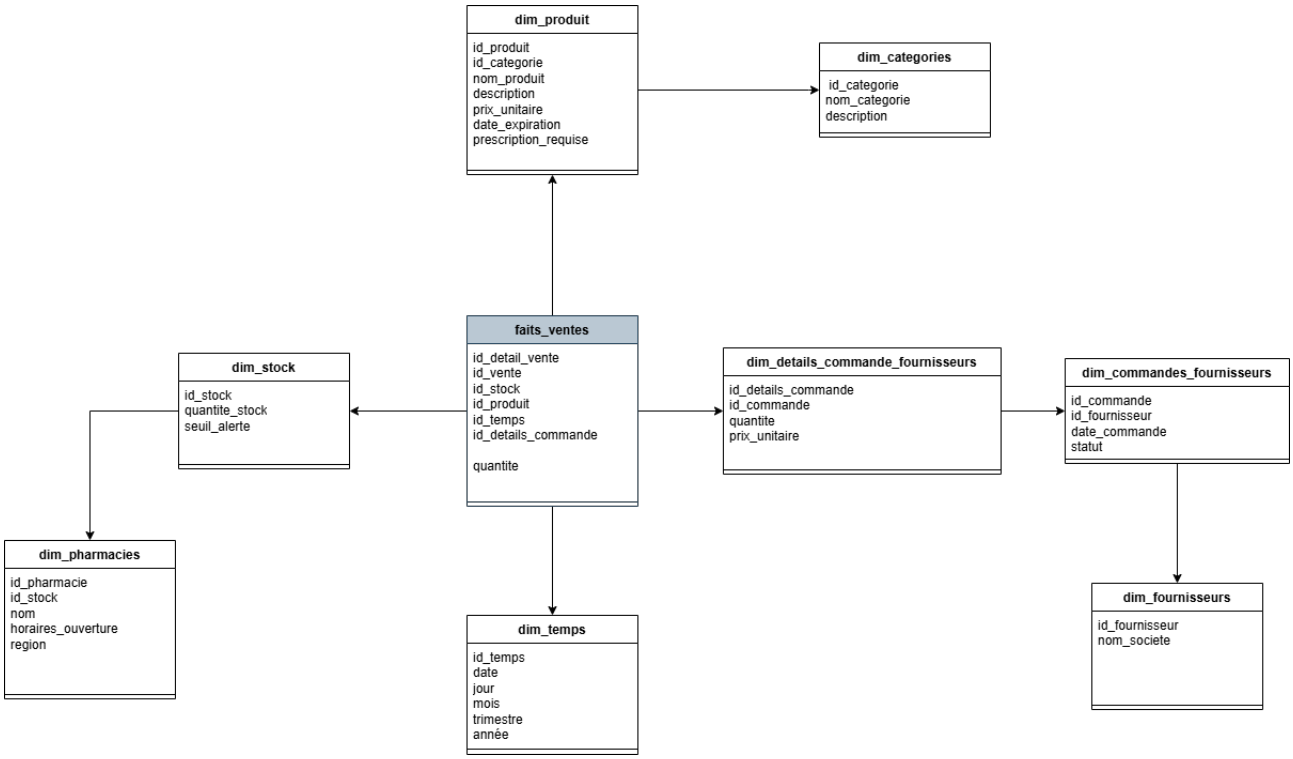
2.1.1 Tables des faits :

1. Faits ventes : Cette table centrale permettra d'analyser les ventes, le stock et les commandes. En situation réelle, il serait préférable de créer trois tables de faits distinctes pour une meilleure modularité et une gestion plus fine des données.
   Cependant, les contraintes de l'énoncé du TP nous obligent à utiliser une seule table de faits (choix entre schéma flocon ou schéma étoile).

2.1.2 Tables des dimensions :

1. Dimension Produits : Contient les informations sur les produits vendus.
2. Dimension Catégories : Classifie les produits par catégories.
3. Dimension Pharmacies : Répertorie les pharmacies concernées.
4. Dimension Stock : Gère les informations relatives au stock disponible de chaque pharmacie.
5. Dimension Temps ventes : Permet l'analyse temporelle des ventes (par jour, mois, année, etc.).
6. Dimension Fournisseurs : Liste les fournisseurs des produits.
7. Dimension commandes fournisseurs : Détaille les commandes passées auprès des fournisseurs.
8. Dimension Details commandes fournisseurs : Fournit des informations spécifiques sur chaque commande.

2.1.3 schéma en Flocon :



2.2 Code SQL pour la création de la BDD OLAP :

```
--Table des faits: faits_ventes
CREATE TABLE faits_ventes (
    id_detail_vente NUMBER NOT NULL,
    id_vente NUMBER NOT NULL,
    id_stock NUMBER ,
    id_produit NUMBER NOT NULL,
    id_temps NUMBER NOT NULL,
    id_detail_commande NUMBER ,
    quantite_vendue NUMBER NOT NULL,
    CONSTRAINT pk_faits_ventes PRIMARY KEY (id_detail_vente
       ,id_vente,id_temps,id_produit),
    CONSTRAINT fk_faits_ventes_stock FOREIGN KEY (id_stock)
        REFERENCES dim_stock(id_stock),
    CONSTRAINT fk_faits_ventes_commande FOREIGN KEY (
       id_detail_commande) REFERENCES
       dim_details_commandes_fr(id_detail_commande),
    CONSTRAINT fk_faits_ventes_produit FOREIGN KEY (
       id_produit) REFERENCES dim_produit(id_produit),
    CONSTRAINT fk_faits_ventes_temps FOREIGN KEY (id_temps)
        REFERENCES dim_temps(id_temps)
);

-- Table dimensionnelle : dim_produit
CREATE TABLE dim_produit (
    id_produit NUMBER PRIMARY KEY,
```

```
    id_categorie NUMBER,
    nom_produit VARCHAR2(100) NOT NULL,
    description VARCHAR2(500),
    prix_unitaire NUMBER(10,2),
    date_expiration DATE,
    prescription_requise NUMBER(1) DEFAULT 0,
    FOREIGN KEY (id_categorie) REFERENCES dim_categories(
       id_categorie)
);

-- Table dimensionnelle : dim_categories
CREATE TABLE dim_categories (
    id_categorie NUMBER PRIMARY KEY,
    nom_categorie VARCHAR2(50) NOT NULL,
    description VARCHAR2(200)
);


-- Table dimensionnelle : dim_stock
CREATE TABLE dim_stock (
    id_stock NUMBER PRIMARY KEY,
    id_pharmacie NUMBER NOT NULL,
    quantite NUMBER DEFAULT 0,
    seuil_alerte NUMBER,
    FOREIGN KEY (id_pharmacie) REFERENCES dim_pharmacies(
       id_pharmacie)
);


-- Table dimensionnelle : dim_pharmacies
CREATE TABLE dim_pharmacies (
    id_pharmacie NUMBER PRIMARY KEY,
    nom VARCHAR2(100) NOT NULL,
    region VARCHAR2(200) NOT NULL,
    horaires_ouverture VARCHAR2(200)
);


-- Table dimensionnelle : dim_temps
CREATE TABLE  DIM_TEMPS
(    ID_TEMPS NUMBER,
    DATE_COMPLETE DATE,
    JOUR NUMBER,
    MOIS NUMBER,
    TRIMESTRE NUMBER,
    ANNEE NUMBER,
     PRIMARY KEY (ID_TEMPS) ENABLE
);

-- Table dimensionnelle : dim_fournisseurs
CREATE TABLE dim_fournisseurs (
    id_fournisseur NUMBER PRIMARY KEY,
    nom_societe VARCHAR2(100) NOT NULL
```

```
);

-- Table dimensionnelle : dim_commandes
CREATE TABLE dim_commandes (
    id_commande NUMBER PRIMARY KEY,
    id_fournisseur NUMBER NOT NULL,
    date_commande DATE,
    statut VARCHAR2(20),
    FOREIGN KEY (id_fournisseur) REFERENCES
       dim_fournisseurs(id_fournisseur)
);


-- Table dimensionnelle : dim_details_commande_fournisseurs
CREATE TABLE dim_details_commandes_fr (
    id_detail_commande NUMBER PRIMARY KEY,
    id_commande NUMBER NOT NULL,
    quantite NUMBER,
    prix_unitaire NUMBER(10,2),
    FOREIGN KEY (id_commande) REFERENCES dim_commandes(
       id_commande)
);
```

2.3 Concevoir le schéma en étoile ou en flocon selon votre analyse :

2.3.1 Comparaison entre schéma en étoile et schéma en flocon :

Critères	**Schéma en Étoile**	**Schéma en Flocon de Neige**
Complexité	Simple et facile à comprendre.	Plus complexe en raison de la décomposition des dimensions en sous-dimensions.
Performances des Requêtes	Optimisé pour les requêtes analytiques rapides grâce à une structure plate.	Moins performant en raison des jointures supplémentaires nécessaires.
Efficacité de Stockage	Présente plus de redondance, ce qui réduit l'efficacité du stockage.	Moins de redondance grâce à la normalisation, ce qui améliore l'efficacité du stockage.
Facilité de Maintenance	Facile à maintenir en raison de sa structure simple.	Maintenance plus complexe en raison de la décomposition des dimensions.

Pour ce TP, un schéma en flocon a été choisi pour les raisons suivantes :

1. Normalisation des dimensions : La décomposition des dimensions "Pharmacies" et "Catégories" permet une granularité plus fine et réduit la redondance des données.
2. Taille des données : Compte tenu du volume de données à traiter, le schéma en flocon offre une meilleure efficacité de stockage tout en maintenant une structure logique et organisée.
3. Objectifs analytiques : Le schéma en flocon permet une analyse plus détaillée et flexible, adaptée aux besoins du TP.

3 Partie 2 : Processus ETL

Avant de démarrer le processus ETL, il est nécessaire de créer les tables correspondant à notre schéma conceptuel de la base de données OLTP. Ces tables serviront de base pour extraire les données fournies avec le TP. Pour cela, un script Pentaho sera utilisé pour automatiser l'extraction, la transformation et le chargement des données.

3.1 Extraction :

3.1.1 extraction des pharmacies :

FIGURE 1 – Extraction des données des pharmacies depuis nos fichier csv

3.1.2 extraction des produits :

FIGURE 2 – Extraction des données des produits depuis nos fichier csv

3.1.3 extraction du stock :

FIGURE 3 – Extraction des données du stock depuis nos fichier csv

3.1.4 extraction des catégories :

FIGURE 4 – Extraction des données des catégories depuis nos fichier csv

3.1.5 extraction des ventes :

FIGURE 5 – Extraction des données des ventes depuis nos fichier csv

3.1.6 extraction des details ventes :

avant l'extraction des données depuis le fichier csv des details ventes j'ai remarqué que id details ventes se repete alors qu'il doit etre unique, alors j'ai créé des identifiants unique pour chaque tuple de details ventes.

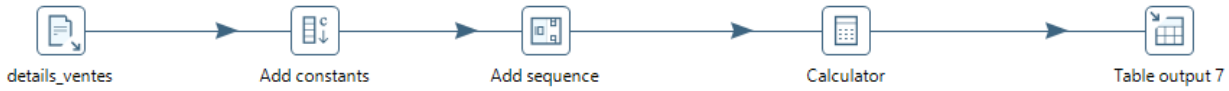


FIGURE 6 – Extraction des données des details ventes depuis nos fichier csv

3.1.7 extraction des fournisseurs :

FIGURE 7 – Extraction des données des fournisseurs depuis nos fichier csv

3.1.8 extraction des commandes fournisseurs :

FIGURE 8 – Extraction des données des commandes fournisseurs depuis nos fichier csv

3.1.9 extraction des details commandes fournisseurs :

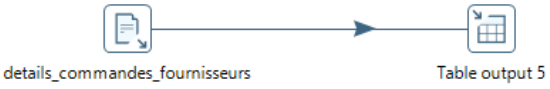


FIGURE 9 – Extraction des données des details commandes fournisseurs depuis nos fichier csv

3.2 Traitement des données :

3.2.1 traitement des pharmacies

On peut remarquer que la table pharmacie dans nos données de base possède un attribut nommé adresse.
Tout d'abord, nous allons traiter cet attribut afin d'extraire la région et de conserver uniquement cette information, en utilisant la requete sql suivante dans le table input du script pentaho :

```
SELECT
    id_pharmacie,
    nom,
    horaires_ouverture,

    CASE
        WHEN SUBSTR(REGEXP_SUBSTR(adresse, '[0-9]{5}'), 1,
           2) BETWEEN '75' AND '95' THEN ' le -de-France'
        WHEN SUBSTR(REGEXP_SUBSTR(adresse, '[0-9]{5}'), 1,
           2) BETWEEN '01' AND '19' THEN 'Nouvelle-
           Aquitaine'
        WHEN SUBSTR(REGEXP_SUBSTR(adresse, '[0-9]{5}'), 1,
           2) BETWEEN '20' AND '39' THEN 'Grand Est'
        WHEN SUBSTR(REGEXP_SUBSTR(adresse, '[0-9]{5}'), 1,
           2) BETWEEN '40' AND '59' THEN 'Hauts-de-France'
        WHEN SUBSTR(REGEXP_SUBSTR(adresse, '[0-9]{5}'), 1,
           2) BETWEEN '60' AND '74' THEN 'Auvergne-Rh ne-
           Alpes'
        WHEN SUBSTR(REGEXP_SUBSTR(adresse, '[0-9]{5}'), 1,
           2) >= '96' THEN 'Outre-Mer'
        ELSE 'Autre'
    END as region
FROM pharmacies
```

ensuite on va simplement introduire cette requete dans notre table

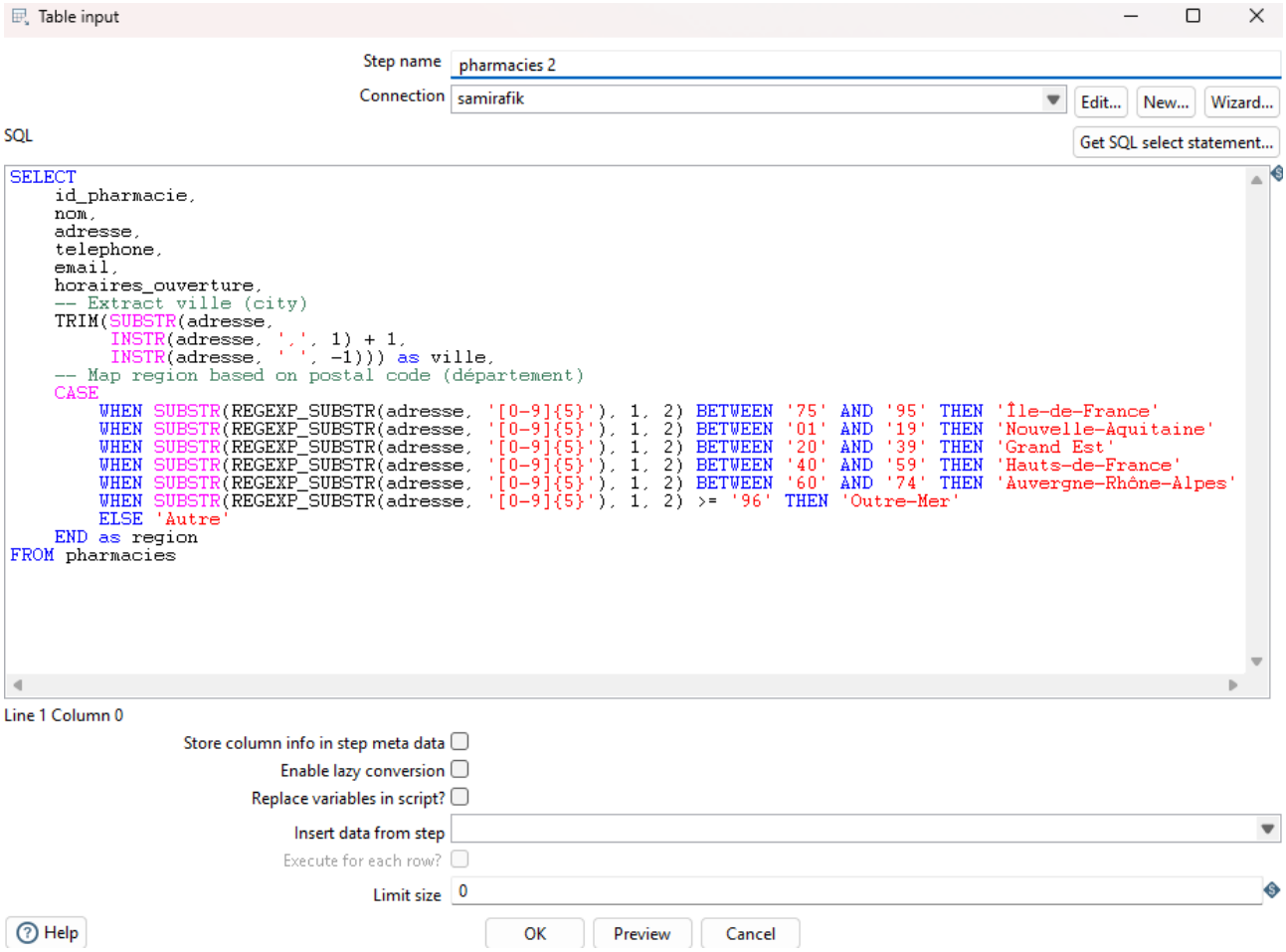


FIGURE 10 – emplacement de la requete dans table input

3.2.2 traitement de details ventes

Après avoir rencontré quelques erreurs lors de la réalisation de ce TP, j'ai constaté qu'une même vente contient des produits dupliqués.
Autrement dit, un vendeur en pharmacie a enregistré un produit deux fois pour une seule et même vente.

investiguation sur details ventes

cette requete SQL permet de numéroter toutes les lignes de 1 et des qu'il trouve un ligne avec le meme id vente et le meme id produit il augemente la numérotation de cette derniere ainsi de suite.

```
SELECT
      dv.*,
      SUM(dv.quantite) OVER (PARTITION BY v.id_vente, dv.
         id_produit) AS total_quantite,
      ROW_NUMBER() OVER (PARTITION BY dv.id_vente,dv.
         id_produit ORDER BY v.id_vente) AS rn
                  FROM ventes v
                  LEFT JOIN details_ventes dv ON dv.
                     id_vente = v.id_vente
```

voici un exemple de duplication :

ID_DETAIL_VENTE	ID_VENTE	ID_PRODUIT	QUANTITE	PRIX_UNITAIRE	TOTAL_QUANTITE	RN
261	47	2	7	130.98	11	1
263	47	2	4	127.97	11	2

FIGURE 11 – produit dupliqué dans une meme vente

une autre requete sql pour lister tous les éléments dupliqués :

```
SELECT v.id_pharmacie, dv.id_produit, v.id_vente
    FROM ventes v
    LEFT JOIN details_ventes dv ON v.id_vente = dv.id_vente
    GROUP BY v.id_pharmacie, dv.id_produit , v.id_vente
    HAVING COUNT(*) > 1
```

et donc on va faire le traitement en selectionnant seulement les tuples qui ont un rn inférieur a égal a 1 grace a ce code SQL qu'on va utilisé par la suite dans notre table input pour remplir la table des faits ventes.

```
SELECT id_detail_vente,
            id_vente,
            id_produit,
            prix_unitaire,
            total_quantite FROM (
                    SELECT
                    dv.*,
                    SUM(dv.quantite) OVER (PARTITION BY v.
                       id_vente, dv.id_produit) AS
                       total_quantite,
                    ROW_NUMBER() OVER (PARTITION BY dv.
                       id_vente,dv.id_produit ORDER BY v.
                       id_vente) AS rn
                    FROM ventes v
                    LEFT JOIN details_ventes dv ON dv.
                       id_vente = v.id_vente
    )
WHERE rn = 1
```

3.2.3 traitement de details du stock

Il existe également des éléments dupliqués dans le stock, ce qui est tout à fait normal pour une même pharmacie. Cependant, en l'absence d'une table permettant de gérer la traçabilité des données, il devient impossible de relier chaque produit d'une vente à son stock associé. Cela entraîne des ambiguïtés, car à chaque tentative, on se retrouve face à des produits dupliqués.

investigation :

```
SELECT
    id_produit,
    id_pharmacie,
    COUNT(*)
FROM Stock
GROUP BY id_produit, id_pharmacie
HAVING COUNT(*) > 1;
```

ID_PRODUIT	ID_PHARMACIE	COUNT(*)
117	9	2
47	19	2
112	14	2
144	8	2
102	15	2
142	4	2
151	1	2
109	2	2
97	7	2

FIGURE 12 – résultat de l'investiguation sur stock

la solution suggérée est de prendre l'id d'un seul stock pour les besoins du TP.
voici la requete sql qu'on va introduire dans notre table input :

```
SELECT
        id_produit,
        id_pharmacie,
        MAX(id_stock) AS id_stock -- Prend le stock le
           plus    lev
    FROM Stock
    GROUP BY id_produit, id_pharmacie
```

3.2.4 traitement de details des details commandes fournisseurs

on rencontre le meme probleme avec les détails commandes fournisseurs

investigation

```
SELECT
    cf.id_pharmacie,
    dcf.id_produit,
    COUNT(*) AS nb_doublons
FROM Details_Commandes_Fournisseurs dcf
INNER JOIN Commandes_Fournisseurs cf
    ON dcf.id_commande = cf.id_commande
GROUP BY cf.id_pharmacie, dcf.id_produit
HAVING COUNT(*) > 1;
```

ID_PHARMACIE	ID_PRODUIT	NB_DOUBLONS
8	128	2
11	26	2
11	171	2
2	41	2
18	183	2
18	38	2
11	71	2
20	4	2
1	11	2
3	105	2
18	193	2
4	178	2

FIGURE 13 – résultat de l'investiguation sur les commandes

pour faciliter notre script pentaho on va créer une table qui agrege les résultats entre les jointures des commandes et des details commandes et prend seulement la commande la plus récente pour les besoins du TP.

```
CREATE table dcf_cf_agregated AS SELECT
    cf.id_pharmacie,
    cf.id_commande,
    dcf.id_produit,
    dcf.id_detail_commande
FROM details_commandes_fournisseurs dcf
INNER JOIN commandes_fournisseurs cf
    ON dcf.id_commande = cf.id_commande
INNER JOIN (
    SELECT
        cf.id_pharmacie,
        dcf.id_produit,
        MAX(dcf.id_detail_commande) AS max_detail_commande
```

```
        FROM details_commandes_fournisseurs dcf
        INNER JOIN commandes_fournisseurs cf
            ON dcf.id_commande = cf.id_commande
        GROUP BY cf.id_pharmacie, dcf.id_produit
    ) agg
        ON dcf.id_detail_commande = agg.max_detail_commande
        AND cf.id_pharmacie = agg.id_pharmacie
        AND dcf.id_produit = agg.id_produit;
```

3.3 Chargement des données dans les tables dimensions et la table des faits ventes

3.3.1 generation de la dimension temps

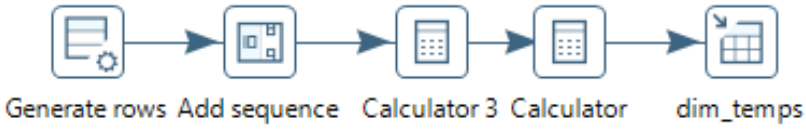


FIGURE 14 – generation des données de la dimension temps

3.3.2 chargement du reste des dimensions :

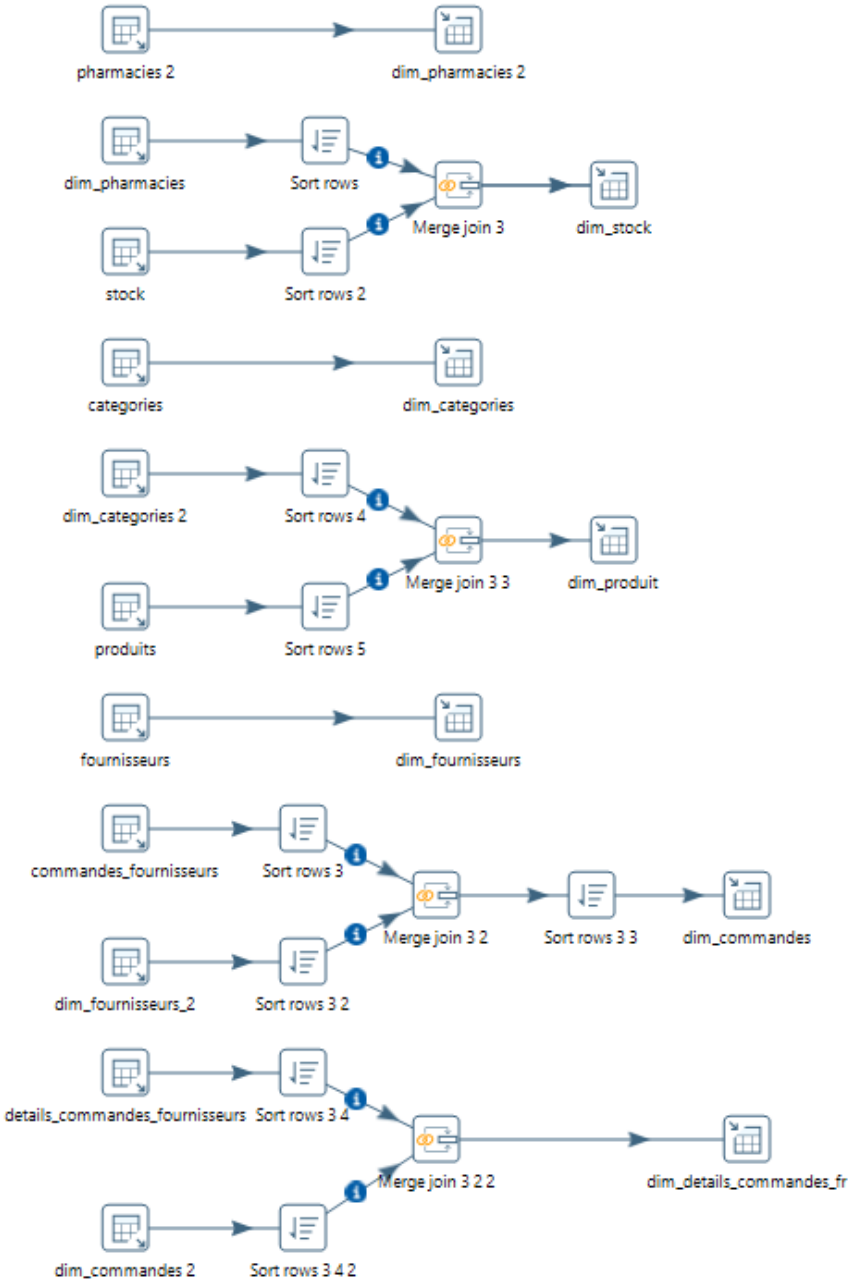


FIGURE 15 – Chargement des dimensions

3.3.3 Chargement de la table des faits ventes :

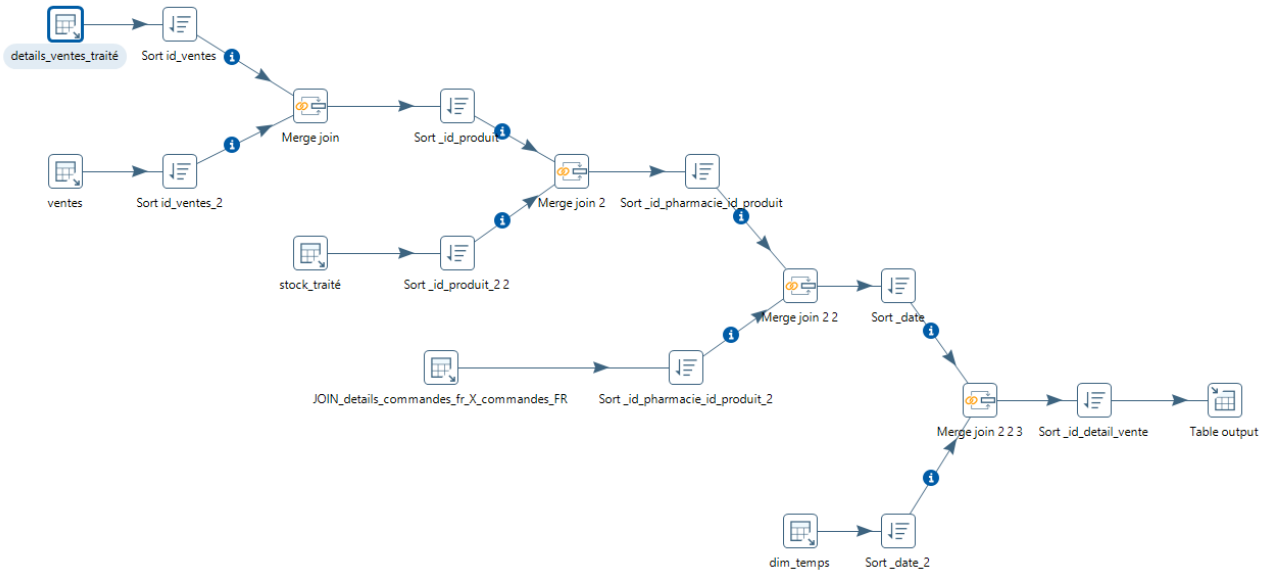


FIGURE 16 – Chargement des données dans la table des faits ventes

Remarque importante : si tout est bon on devrait avoir 5911343 instances dans notre table des faits sinon le TP est faux.

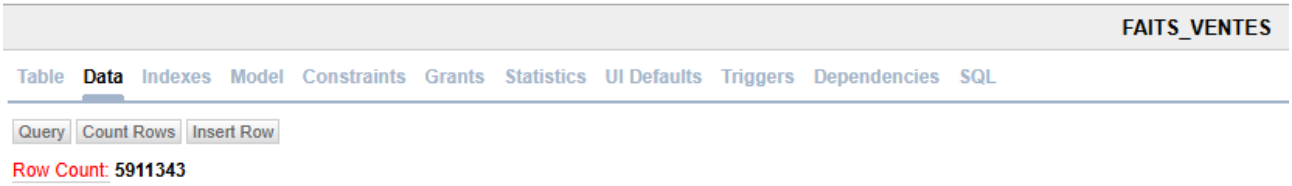


FIGURE 17 – nombre d'instances dans faits ventes

nos données devraient alors ressembler a ça :

ID_DETAIL_VENTE	ID_VENTE	ID_STOCK	ID_PRODUIT	ID_TEMPS	ID_DETAIL_COMMANDE	QUANTITE_VENDUE
1167	198	-	57	431	-	8
1168	199	-	15	436	-	7
1169	199	-	152	436	-	9
1170	199	-	99	436	-	9
1171	199	267	1	436	-	8
1172	200	-	107	548	-	7
1173	200	490	140	548	-	2
1174	200	-	192	548	-	3
1175	200	-	8	548	-	5
1176	200	-	199	548	299	4
1177	200	-	144	548	-	3
1178	200	352	176	548	20	3
1179	201	-	194	567	-	5
1180	201	287	145	567	-	9
1181	201	-	170	567	-	2
1182	201	-	82	567	-	2

FIGURE 18 – échantillon de faits ventes

4 stratégie d'historisation des données

4.1 Dimensions à Évolution Lente :

Exemples : `dim_pharmacies`, `dim_categories`, `dim_fournisseurs`

Approche

- Ajout de champs temporels :
  - `date_debut` (DATE)
  - `date_fin` (DATE)
  - `version` (NUMBER)
  - `is_current` (NUMBER(1))
- Conservation des anciennes versions avec mise à jour des dates

Listing 1 – Exemple pour dim_produit

```
ALTER TABLE dim_produit ADD (
    date_debut DATE DEFAULT SYSDATE,
    date_fin DATE DEFAULT TO_DATE('31-12-2025', 'DD-MM-YYYY'),
    version NUMBER DEFAULT 1,
    is_current NUMBER(1) DEFAULT 1
);
```

Avantages

Historisation	Conservation précise de l'historique sans perte de données
Compatibilité	Intégration transparente avec les requêtes temporelles via `dim_temps`

4.2 Dimensions à Évolution Rapide :

Exemples :

- `dim_stock` (attribut `quantite`)
- `dim_details_commandes_fr` (attribut `prix_unitaire`)

Approche

- Création de tables d'historique (mouvement)
- Liaison via clés étrangères supplémentaires

Listing 2 – Exemple pour dim_stock

```
CREATE TABLE hist_stock (
    id_hist_stock NUMBER PRIMARY KEY,
    id_stock NUMBER,
    quantite NUMBER,
    date_modification DATE,
    FOREIGN KEY (id_stock) REFERENCES dim_stock(id_stock)
);
```

Avantages

Performance	Évite la surcharge de versions dans la table principale
Maintenance	Optimise la gestion des attributs fréquemment modifiés

5 Partie 3 : Requêtes Analytiques

5.1 Analyser l'évolution des ventes mensuelles par catégorie de produits

```
SELECT
  dt.annee,
  dt.mois,
  dc.nom_categorie,
  dp.nom_produit,
  SUM(fv.quantite_vendue) AS total_ventes,

  CASE GROUPING_ID(dt.annee, dt.mois, dc.nom_categorie, dp.
     nom_produit)
    WHEN 1 THEN 'Total Produit'
    WHEN 3 THEN 'Total Cat gorie'
    WHEN 7 THEN 'Total Mensuel'
    WHEN 15 THEN 'Total Global'
    ELSE 'Detail'
  END AS niveau_agregation
FROM faits_ventes fv
JOIN dim_temps dt ON fv.id_temps = dt.id_temps
JOIN dim_produit dp ON fv.id_produit = dp.id_produit
JOIN dim_categories dc ON dp.id_categorie = dc.id_categorie
GROUP BY
  ROLLUP(dt.annee, dt.mois),
  CUBE(dc.nom_categorie, dp.nom_produit)
ORDER BY
  dt.annee,
  dt.mois,
  GROUPING(dt.annee),  -- Tri des totaux en dernier
  GROUPING(dc.nom_categorie),
  dc.nom_categorie,
  dp.nom_produit;
```

ANNEE	MOIS	NOM_CATEGORIE	NOM_PRODUIT	TOTAL_VENTES	NIVEAU_AGREGATION
2023	12	Aromathérapie	Huile essentielle de lavande	43970	Détail
2023	12	Aromathérapie	Huile essentielle de tea tree	75005	Détail
2023	12	Aromathérapie	-	118975	Total Produit
2023	12	Bébé	Crème pour le change	32188	Détail
2023	12	Bébé	Sérum physiologique	87342	Détail
2023	12	Bébé	-	119530	Total Produit
2023	12	Contraception	Malox Sirop	86458	Détail
2023	12	Contraception	Test de grossesse	53731	Détail
2023	12	Contraception	-	140189	Total Produit
2023	12	Cosmétiques	Démaquillant	129281	Détail
2023	12	Cosmétiques	Eau micellaire	128821	Détail
2023	12	Cosmétiques	-	258102	Total Produit
2023	12	Dermatologie	Crème hydratante	10617	Détail
2023	12	Dermatologie	Sérum visage	10211	Détail
2023	12	Dermatologie	Écran solaire	42846	Détail
2023	12	Dermatologie	-	63674	Total Produit

FIGURE 19 – échantillon du résultat d'analyse de l'évolution des ventes mensuelles par catégorie de produits

5.2 Calculer le taux de rotation des stocks par pharmacie

1. **Taux de rotation des stocks** :

$$\text{Taux de rotation des stocks} = \frac{\text{Coût des ventes}}{\text{Stock moyen}}$$

2. **Coût des ventes ou des consommations** :

$$\text{Coût des ventes} = \text{Valeur totale des médicaments ou produits vendus}$$

3. **Stock moyen** :

$$\text{Stock moyen} = \frac{\text{Stock initial} + \text{Stock final}}{2}$$

4. **Stock final** :

$$\text{Stock final} = \text{Stock initial} + \text{Quantité achetée} - \text{Quantité vendue}$$

```
        SELECT
            fv.id_produit,
            SUM(s.quantite) AS "quantite initiale",
            SUM(ddc.quantite) AS "quantite achetee",
            SUM(fv.quantite_vendue) AS "quantite vendue",
            (SUM(s.quantite) + SUM(ddc.quantite) - SUM(fv.
               quantite_vendue)) AS "stock final",
            (SUM(s.quantite) + (SUM(ddc.quantite) + SUM(s.quantite) -
               SUM(fv.quantite_vendue)) / 2) AS "stock moyen",
            -- Calcul de la rotation : (Quantite vendue / Stock moyen)
            CASE
                WHEN (SUM(s.quantite) + (SUM(ddc.quantite) + SUM(s.
                   quantite) - SUM(fv.quantite_vendue)) / 2) = 0
                THEN 0  -- Evite la division par zero
```

```
            ELSE ROUND(SUM(fv.quantite_vendue*dp.prix_unitaire) / (
               SUM(s.quantite) + (SUM(ddc.quantite) + SUM(s.
               quantite) - SUM(fv.quantite_vendue)) / 2), 2)
        END AS "rotation"
    FROM dim_stock s
    LEFT JOIN faits_ventes fv ON s.id_stock = fv.id_stock
    LEFT JOIN dim_details_commandes_fr ddc ON ddc.
       id_detail_commande = fv.id_detail_commande
    LEFT JOIN dim_commandes dc ON dc.id_commande = ddc.id_commande
    LEFT JOIN dim_produit dp ON fv.id_produit = dp.id_produit
    GROUP BY fv.id_produit
    ORDER BY fv.id_produit ASC;
```

ID_PRODUIT	quantité initiale	quantité achetée	quantité vendue	stock final	stock moyen	rotation
1	1628130	-	33381	-	-	-
2	778140	-	8056	-	-	-
3	1178060	-	8343	-	-	-
4	1507640	-	16841	-	-	-
5	3033680	-	33193	-	-	-
6	864633	566181	8205	1422609	1575937.5	.22
7	1292106	-	24743	-	-	-
8	982128	-	16924	-	-	-
9	1107148	-	17002	-	-	-
10	1172847	-	8363	-	-	-
11	2997962	-	41110	-	-	-
12	463680	-	8260	-	-	-
13	2062304	-	16488	-	-	-
15	1455267	-	24599	-	-	-
16	1426569	-	16557	-	-	-
17	2171171	58880	32242	2197809	3270075.5	.06
18	2671836	-	24776	-	-	-
19	481185	-	8166	-	-	-
20	221861	198037	8237	411661	427691.5	1.4
21	641563	-	16169	-	-	-
22	1361520	-	8056	-	-	-
23	3246126	-	42037	-	-	-
24	1333877	366102	16108	1683871	2175812.5	1.09

FIGURE 20 – échantillon du résultat de la rotation du stock

5.3 Identifier les produits les plus rentables par région

```sql
SELECT
  COALESCE(p.region, 'Total General') AS region,
  COALESCE(dp.nom_produit, 'Total Region') AS produit,
  SUM(fv.quantite_vendue * dp.prix_unitaire) AS
     chiffre_affaires,
  CASE GROUPING_ID(p.region, dp.nom_produit)
    WHEN 1 THEN 'Total Produit'
    WHEN 3 THEN 'Total Global'
    ELSE 'Detail'
  END AS niveau_agregation
FROM faits_ventes fv
JOIN dim_produit dp ON fv.id_produit = dp.id_produit
JOIN dim_stock st ON fv.id_stock = st.id_stock
JOIN dim_pharmacies p ON st.id_pharmacie = p.id_pharmacie
GROUP BY ROLLUP(p.region, dp.nom_produit)
ORDER BY region, GROUPING(dp.nom_produit), chiffre_affaires
   DESC;
```

REGION	PRODUIT	CHIFFRE_AFFAIRES	NIVEAU_AGREGATION
Auvergne-Rhône-Alpes	Eau micellaire	4393400.55	Détail
Auvergne-Rhône-Alpes	Malox Sirop	4362447.27	Détail
Auvergne-Rhône-Alpes	Solution pour lentilles	4051463.1	Détail
Auvergne-Rhône-Alpes	Démaquillant	3631120.35	Détail
Auvergne-Rhône-Alpes	Calcium	3060102.44	Détail
Auvergne-Rhône-Alpes	Mélatonine	2493453.84	Détail
Auvergne-Rhône-Alpes	Sérum physiologique	1871301.78	Détail
Auvergne-Rhône-Alpes	Huile essentielle de lavande	1776356.81	Détail
Auvergne-Rhône-Alpes	Calendula	1617965.85	Détail
Auvergne-Rhône-Alpes	Fer	1558497.27	Détail
Auvergne-Rhône-Alpes	Multivitamines	1493385.75	Détail
Auvergne-Rhône-Alpes	Doliprane	1436453.61	Détail
Auvergne-Rhône-Alpes	Gel douche	1352937.6	Détail
Auvergne-Rhône-Alpes	Béquilles	1334307.09	Détail
Auvergne-Rhône-Alpes	Arnica	1124592.51	Détail
Auvergne-Rhône-Alpes	Kit de premiers secours	962953.66	Détail
Auvergne-Rhône-Alpes	Lingettes nettoyantes	895190.96	Détail
Auvergne-Rhône-Alpes	Ceinture lombaire	836966.68	Détail
Auvergne-Rhône-Alpes	Ibuprofen	725076.24	Détail
Auvergne-Rhône-Alpes	Huile essentielle de tea tree	716622.4	Détail

FIGURE 21 – échantillon du résultat des produits les plus rentables par région

5.4 Analyser les tendances saisonnières des ventes par catégorie

On suppose qu'une saison est un trimestre.

```
SELECT
  COALESCE(dc.nom_categorie, 'Toutes categories') AS categorie,
  COALESCE(TO_CHAR(dt.trimestre), 'Tous trimestres') AS
     trimestre,
  dt.annee,
  SUM(fv.quantite_vendue) AS total_vendu,
  CASE
    WHEN GROUPING(dc.nom_categorie) = 1 THEN 'Total Trimestre'
    WHEN GROUPING(dt.trimestre) = 1 THEN 'Total Categorie'
    ELSE 'Detail'
  END AS aggregation_level
FROM faits_ventes fv
JOIN dim_temps dt ON fv.id_temps = dt.id_temps
JOIN dim_produit dp ON fv.id_produit = dp.id_produit
JOIN dim_categories dc ON dp.id_categorie = dc.id_categorie
GROUP BY CUBE(dc.nom_categorie), ROLLUP(dt.annee, dt.trimestre)
ORDER BY dt.annee, dt.trimestre;
```

CATEGORIE	TRIMESTRE	ANNEE	TOTAL_VENDU	AGGREGATION_LEVEL
Aromathérapie	4	2023	118975	Detail
Premiers secours	4	2023	97062	Detail
Contraception	4	2023	140189	Detail
Médicaments	4	2023	108062	Detail
Homéopathie	4	2023	215637	Detail
Diététique	4	2023	107549	Detail
Dermatologie	4	2023	63674	Detail
Cosmétiques	4	2023	258102	Detail
Orthopédie	4	2023	141097	Detail
Relaxation	4	2023	85959	Detail
Nutrition	4	2023	152576	Detail
Hygiène	4	2023	140474	Detail
Optique	4	2023	215928	Detail
Bébé	4	2023	119530	Detail
Toutes categories	4	2023	2160034	Total Trimestre
Parapharmacie	4	2023	195220	Detail
Toutes categories	Tous trimestres	2023	2160034	Total Trimestre

FIGURE 22 – échantillon du résultat des tendances saisonnières des ventes par catégorie

6 les optimisations :

6.1 Index des Clés Étrangères

```
CREATE INDEX idx_fv_stock ON faits_ventes(id_stock);
CREATE INDEX idx_fv_commande ON faits_ventes(id_detail_commande);
CREATE INDEX idx_fv_produit ON faits_ventes(id_produit);
CREATE INDEX idx_fv_temps ON faits_ventes(id_temps);
```

Justification : Améliore les jointures entre `faits_ventes` et les tables dimensionnelles.

6.2 dim_produit

```
CREATE INDEX idx_produit_categorie ON dim_produit(id_categorie);
CREATE INDEX idx_produit_nom ON dim_produit(nom_produit);
```

6.3 dim_stock

```
CREATE INDEX idx_stock_pharmacie ON dim_stock(id_pharmacie);
```

6.4 dim_commandes

```
CREATE INDEX idx_commande_fournisseur ON dim_commandes(id_fournisseur);
```

6.5 dim_details_commandes_fr

```
CREATE INDEX idx_details_commande ON dim_details_commandes_fr(
   id_commande);
```

6.6 dim_temps

```
CREATE INDEX idx_temps_date ON dim_temps(date_complete);
```

6.7 dim_pharmacies

```
CREATE INDEX idx_pharmacie_region ON dim_pharmacies(region);
```

6.8 faits_ventes + dim_produit

```
CREATE INDEX idx_fv_produit_quantite ON faits_ventes(id_produit,
   quantite_vendue);
```

6.9 dim_produit + dim_categories

```
CREATE INDEX idx_produit_cat_prix ON dim_produit(id_categorie,
   prix_unitaire);
```

6.10 faits_ventes + dim_temps

```
CREATE INDEX idx_fv_temps_quantite ON faits_ventes(id_temps,
   quantite_vendue);
```

Justification des Choix

Index	Table	Bénéfice
idx_fv_stock	faits_ventes	Optimise les jointures avec dim_stock
idx_produit_categorie	dim_produit	Accélère les jointures avec dim_categories
idx_temps_date	dim_temps	Améliore les requêtes filtrées par date
idx_pharmacie_region	dim_pharmacies	Optimise les analyses régionales
idx_fv_temps_quantite	faits_ventes	Rend les agrégations temporelles plus rapides

6.11 Conclusion

Cette stratégie d'indexation permet d'optimiser significativement les temps d'exécution des requêtes tout en conservant une maintenance raisonnable. Une surveillance régulière des performances est recommandée pour ajuster les index en fonction de l'évolution des données.