
# Mini-projet HMIN122M

## EDBD : Entrepôts de Données

---

## Master 1 Informatique

## Parcours IMAGINA / DECOL

---

Cédric PLUTA (20124156)
Elodie TRIBOUT (20142504)
Natdanaï BESSON (20135753)

Année Universitaire 2018 - 2019

# Table des matières

# 1 Développement d'un Data Warehouse fictif pour l'entreprise Décathlon

## I. Introduction

Les entrepôts de données permettent de répondre aux besoins précis des entreprises et leur permettent de tirer le plus d'informations possibles afin d'affiner leurs stratégies de déploiement. La mise en place d'un entrepôt de données dépend donc exclusivement des traitements permettant aux entreprises d'améliorer leur position dans le marché (e.g., la vente d'un produit) ou encore la logistique et la gestion de son fonctionnement interne (e.g., les inventaires, la chaîne de production).

Il est donc impératif de ne pas stocker dans un entrepôt toutes les données disponibles dans les différentes sources (sauf pour des cas rares), et de l'alimenter seulement avec les données nécessaires aux requêtes analytiques.

En outre, vu le coût de mise en place d'un système d'aide à la décision, il est obligatoire de modéliser avec une grande précision les actions / opérations dont l'analyse est la plus rentable (e.g., augmenter les ventes en proposant des promotions de produit pour une population ciblée).

C'est dans le cadre de l'exploitation et l'application de nos connaissances acquises dans l'UE HMIN122M *Entrepôt de Données et Big Data* que nous réalisons ce mini-projet sur l'enseigne française Décathlon.

# II. Analyse des besoins

L'enseigne Décathlon est le leader européen du commerce de détail des articles de sport. Afin d'assurer cette position sur ce marché qui malgré tout stagne face à la multiplication de la concurrence et de la diminution des achats de "renouvellement", la direction souhaite développer différentes stratégies de développement et d'innovation grâce à l'analyse des différentes opérations ayant lieu dans leurs magasins.

## A. Opérations importantes du sujet

Nous nous sommes donc mis à la place d'un responsable de direction de l'enseigne Decathlon désirant obtenir un outil d'aide à la décision. Pour ce faire, il faut d'abord déterminer quels sont les besoins de son entreprise.

- Développement et amélioration des marques distributeurs Décathlon
- Amélioration du ciblage marketing
- Optimisation du rayonnage des magasins
- Augmentation du volume des ventes
- Optimisation de la gestion des stocks
- Amélioration du système de livraison
- Optimisation de la gestion des ressources humaines

## B. Traitements possibles

- Développement et amélioration des marques distributeurs Décathlon
  - Quelles sont les produits de marque distributeur Décathlon qui se vendent moins bien que les autres marques proposées par les magasins ?
  - Quels sont les types de produits que les marques distributeurs Décathlon ne possèdent pas et qui se vendent le plus ?
  - Quels sont les produits des marques distributeurs "Décathlon" avec le plus de retours au Service Après-Vente ?
- Amélioration du ciblage marketing
  - Quels sont les types d'articles les plus vendus sur une période donnée par des clients enregistrés au programme fidélité ?
  - Quels sont les sports les plus pratiqués par magasin en fonction des achats ?
  - Quel est le montant total d'achats par membre du programme fidélité ?
- Optimisation du rayonnage des magasins
  - Quels sont les équipements de sport qui rapportent le moins ?
  - Quels sont les types de produits les plus souvent achetés ensembles ?
  - Quels sont les produits les plus vendus par saison et par magasin ?
- Augmentation du volume des ventes
  - Chiffre d'affaire total en fonction de la saison par magasin ?
  - Quel est la quantité de produit vendu par catégorie et par magasin ?
  - Quels sont les types de produits vendus par saison et par magasin ?
- Optimisation de la gestion des stocks
  - Quels sont les magasins ayant le plus de produits en stock à un jour donné ?
  - Quelle est la quantité restante pour chaque produit par magasin ?
  - Quels sont les produits dont la quantité est inférieure à un certain seuil ?
- Amélioration du système de livraison

  - Quelles sont les zones géographiques où il y a le plus de livraison ?
  - Quelle est la moyenne de livraison par jour et par ville ?
  - Quel est le nombre de retrait en magasin par semaine ?
- Optimisation de la gestion des ressources humaines
  - Quelle est la proportion de "retrait en magasin" suite à une commande en ligne par magasin sur une période donnée ?
  - Quel est le nombre de produits vendu par sport, par tranche horaire et par magasin ?
  - Quels sont les types de produits vendus par saison et par magasin ?

### C. Ordonnancement des actions par ordre de priorité

1. Augmentation du volume des ventes
2. Optimisation de la gestion des stocks
3. Amélioration du ciblage marketing
4. Développement et amélioration les marques distributeurs Décathlon
5. Optimisation du rayonnage des magasins
6. Amélioration du système de livraison
7. Optimisation de la gestion des ressources humaines

Nous pensons, hypothétiquement, qu'à travers la conception d'un Data Warehouse, l'entreprise Décathlon ne va pas se focaliser uniquement sur la réduction éventuelle des coûts mais sur l'optimisation des dépenses en incluant leurs fournisseurs. Elle veut alors pouvoir prévoir les commandes fournisseur à l'avance grâce aux tendances obtenues afin d'éviter les ruptures de stock et positionner les marchandises dans les zones plus demandeuses et ainsi réduire les temps d'acheminement pour le réapprovisionnement.

On obtient alors une augmentation des ventes par une plus grande finesse dans la réponse aux besoins des clients et une réduction des coûts par les optimisations de stocks. Décathlon va ainsi chercher à agir de telle sorte qu'elle puisse continuer son développement stratégique.

L'ordre donné est donc en fonction de la meilleure rentabilité potentielle.

# III. Conception de l'Entrepôt de Données

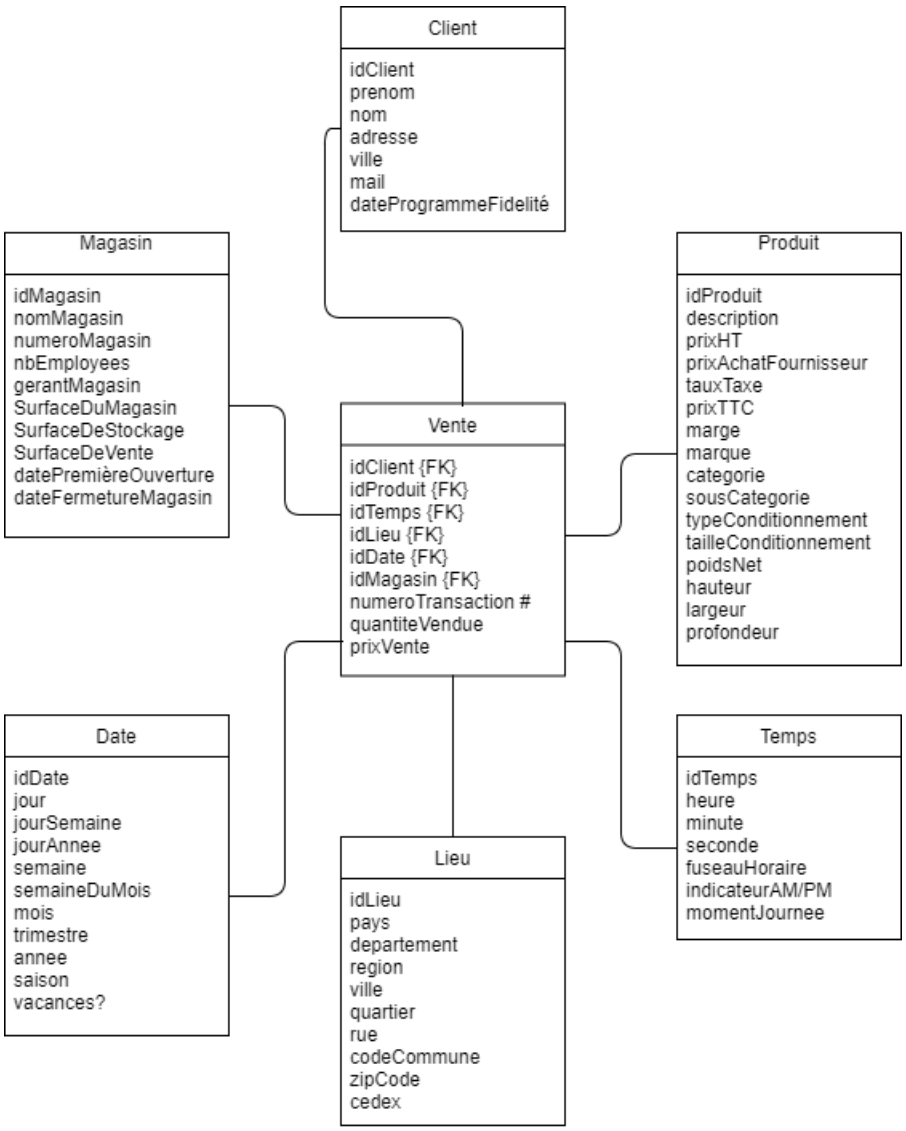
## A. Opérations les plus importantes à analyser

Les opérations les plus importantes à analyser seront donc l'augmentation du volume des ventes ainsi que l'optimisation de la gestion des stocks afin d'avoir à priori une augmentation du chiffre d'affaire et une réduction des coûts d'exploitation.

## B. Conception des Data-Marts

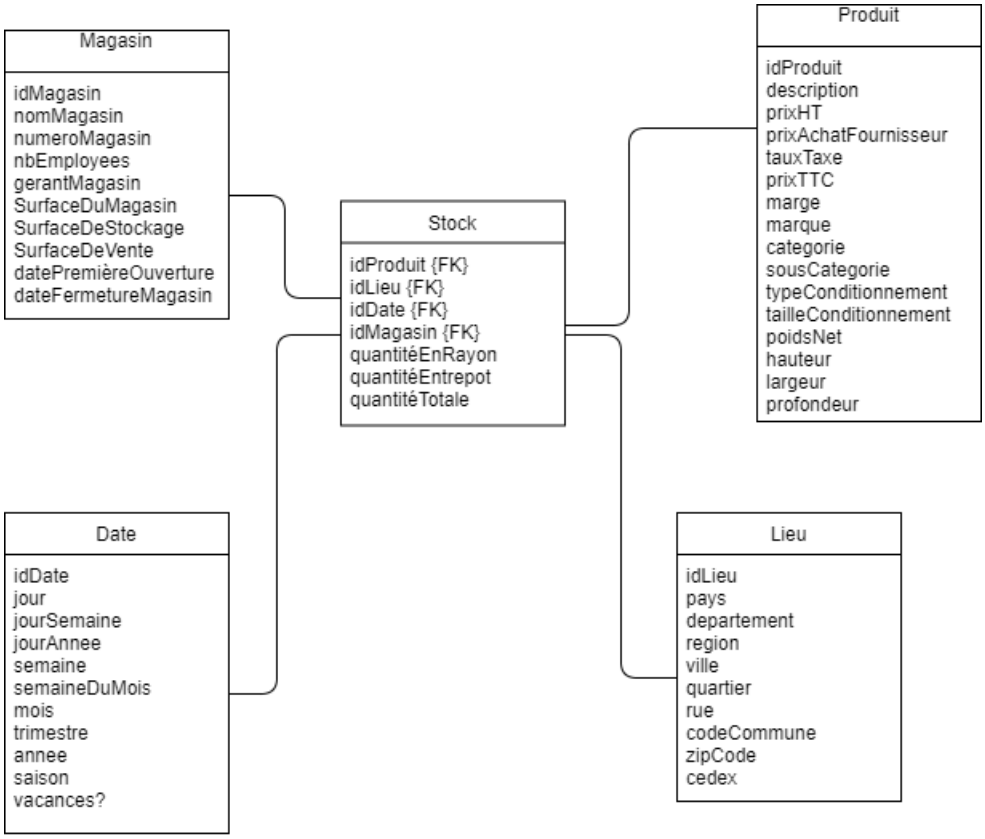
### Augmentation du volume des ventes

La base de faits pour ce Data-Mart se nomme 'Vente' et sera de type transactionnelle avec des mesures semi-additives.



## Optimisation de la gestion des stocks

La base de fait de ce Data-Mart, issu d'un snapshot périodique, se nomme 'Stock' et la mesure sera semi-additive. Elle est semi-additive car l'on va additionner selon certaines dimensions seulement et pas toutes.




Dans le rapport précédent, nous avions intégré une dimension supplémentaire, "Livraison" qui finalement n'était pas cohérente avec notre représentation du problème de la gestion des stocks. À cela, nous l'avons supprimé. Nous avons par la suite ajouté une dimension "Lieu" qui permettra d'affiner les recherches en fonction de zones géographiques particulières (par pays par exemple)

## Définition des dimensions nécessaires pour chaque opération

### Augmentation du volume des ventes

- Vente(***idClient*, *idProduit*, *idTemps*, *idLieu*, *idDate*, *idMagasin***, numeroTransaction, quantiteVendue, prixVente)
- Client(**idClient**, prenom, nom, adresse, ville, mail, dateProgrammeFidelite)
- Produit(**idProduit**, description, prixHT, prixAchatFournisseur, tauxTaxe, prixTTC, marge, marque, categorie, sousCategorie, typeConditionnement, tailleConditionnement, poidsNet, hauteur, largeur, profondeur)
- Temps(**idTemps**, heure, minute, seconde, fuseauHoraire, indicateurAMPM, momentJournee)
- Lieu(**idLieu**, pays, departement, region, ville, quartier, rue, codeCommune, zipCode, cedex)
- CurrentDate(**idDate**, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois, trimestre, annee, saison, vacances)
- Magasin(**idMagasin**, nomMagasin, numeroMagasin, nbEmployees, gerantMagasin, SurfaceDuMagasin, SurfaceDeStockage, SurfaceDeVente, datePremièreOuverture, dateFermetureMagasin)

### Optimisation de la gestion des stocks

- Stock(***idProduit*, *idLieu*, *idDate*, *idMagasin***, quantiteEnRayon, quantiteEntrepot, quantiteTotale)
- Produit(**idProduit**, description, prixHT, prixAchatFournisseur, tauxTaxe, prixTTC, marge, marque, categorie, sousCategorie, typeConditionnement, tailleConditionnement, poidsNet, hauteur, largeur, profondeur)
- Lieu(**idLieu**, pays, departement, region, ville, quartier, rue, codeCommune, zipCode, cedex)
- CurrentDate(**idDate**, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois, trimestre, annee, saison, vacances)
- Magasin(**idMagasin**, nomMagasin, numeroMagasin, nbEmployees, gerantMagasin, SurfaceDuMagasin, SurfaceDeStockage, SurfaceDeVente, datePremiereOuverture, dateFermetureMagasin)

## Le modèle est-il adapté aux traitements de ces actions ?

### Augmentation du volume des ventes

- Chiffre d'affaire total en fonction de la saison par magasin ? OUI
- Quel est la quantité de produit vendu par catégorie et par magasin ? OUI
- Quel est le total des ventes par magasin et par type de produit ? OUI

Notre modèle permet de répondre aux traitements des opérations principales que nous avons retenues.

### Optimisation de la gestion des stocks

- Quels sont les magasins ayant le plus de produits en stock à un jour donné ? OUI
- Quelle est la quantité restante pour chaque produit par magasin ? OUI
- Quels sont les produits dont la quantité est inférieure à un certain seuil ? OUI

Notre modèle permet de répondre aux traitements des opérations principales que nous avons retenues.

## Instanciation du modèle

### Augmentation du volume des ventes :

**Magasin :**
(1, "Montpellier-Odysseum", 157, 199, "Jean-Michel Amoitier", 3900, 1000, 2600, "2008/09/01", NULL)
(2, "Quetigny-Dijon", 82, 186, "Jean-Michel Apeupres", 3500, 1000, 2200, "2009/02/11", NULL)
(3, 'Orange', 329, 198, "Jean-Michel Padevoix", 3900, 1200, 2600, "2007/09/01", NULL)

**CurrentDate :**
(1, 29, 1, 302, 43, 5, 10, 4, 2018, 'Automne', 1)
(2, 30, 2, 303, 43, 5, 10, 4, 2018, 'Automne', 1)
(3, 31, 3, 304, 43, 5, 10, 4, 2018, 'Automne', 1)

**Lieu :**
(1, 'France', 'Herault', 'Occitanie', 'Montpellier', 'Millenaire', "1072 Avenue Georges Melies", 34172, 34000, 4)
(2, 'France', "Cote d'Or", "Bourgogne-Franche-Comte", 'Quetigny', "ZI Quetigny", "Rue Champeau", 21515, 21800, NULL)
(3, 'France', 'Vaucluse', "Provence-Alpes-Cote-D'Azur", 'Orange', "ZA Porte Sud", "ZA Porte Sud", 84087, 84100, NULL)

**Temps :**
(1, 2, 40, 29, "+01", 'PM', "Apres-Midi")
(2, 3, 16, 47, "+01", 'PM', "Apres-Midi")
(3, 11, 48, 56, "+01", 'AM', 'Matin')
(4, 3, 33, 9, "+01", 'PM', "Apres-Midi")

**Produit :**
(1, "Expert TR 960", 80.00, 55.59, 20, 96, 24.41, 'ARTENGO', 'Tennis', "Raquette Tennis Adulte", 'libre', NULL, 0.360, 86.5, 35, 2.8)
(2, "Kiprun Trail Jaune taille 40", 50.00, 29.9, 20, 60, 20.10, 'KALENJI', 'Trail', "Chaussure de Trail Homme", 'libre', NULL, 0.390, 7.5, 9.8, 26.5)
(3, "Kiprun Trail Grise taille 39", 50, 28.55, 20, 60, 21.45, 'KALENJI', 'Trail', "Chaussure de Trail Femme", 'libre', NULL, 0.38, 7.5, 9.8, 25.5)

(4, "Survetement 560 Gym Bleu 10 ans", 4.8, 3.5, 20, 6, 1.3, 'DOMYOS', 'Gym', "Survetement Fille", 'libre', NULL, 0.49, 56, 36, 15)

**Client :**
(1, 'Michel', 'Bergeait', "55 Rue du Parvis", 'Grabels', "michel.berger@mail.fr", "2009/08/11")
(2, NULL, NULL, NULL, 'Quetigny', "david.goodenough@mail.fr", NULL)
(3, 'Henri', 'Lechanteur', "256 rue des oiseaux", 'Orange', "henri.cestmoi@mail.fr", "2014/03/15")
(4, 'David', 'Goodenought', "40 quai des Carrosses", 'Montpellier', "matuture34@mail.fr", "2018/10/30")

**Vente :**
(1, 1, 1, 1, 1, 1, 2, 192.00)
(2, 2, 2, 2, 1, 2, 2, 1, 60.00)
(3, 3, 3, 3, 2, 3, 3, 1, 60.00)
(4, 4, 4, 1, 3, 1, 4, 1, 6.00)

### Optimisation de la gestion des stocks :

**Magasin :**
(1, "Montpellier-Odysseum", 157, 199, "Jean-Michel Amoitier", 3900, 1000, 2600, "2008/09/01", NULL)
(2, "Quetigny-Dijon", 82, 186, "Jean-Michel Apeupres", 3500, 1000, 2200, "2009/02/11", NULL)
(3, 'Orange', 329, 198, "Jean-Michel Padevoix", 3900, 1200, 2600, "2007/09/01", NULL)

**CurrentDate :**
(4, 1, 4, 305, 43, 1, 11, 4, 2018, 'Automne', 1)
(5, 2, 5, 306, 43, 1, 11, 4, 2018, 'Automne', 1)
(6, 3, 6, 307, 43, 1, 11, 4, 2018, 'Automne', 1)

**Lieu :**
(1, 'France', 'Herault', 'Occitanie', 'Montpellier', 'Millenaire', "1072 Avenue Georges Melies", 34172, 34000, 4)
(2, 'France', "Cote d'Or", "Bourgogne-Franche-Comte", 'Quetigny', "ZI Quetigny", "Rue Champeau", 21515, 21800, NULL)
(3, 'France', 'Vaucluse', "Provence-Alpes-Cote-D'Azur", 'Orange', "ZA Porte Sud", "ZA Porte Sud", 84087, 84100, NULL)

**Produit :**
(1, "Expert TR 960", 80.00, 55.59, 20, 96, 24.41, 'ARTENGO', 'Tennis', "Raquette Tennis Adulte", 'libre', NULL, 0.360, 86.5, 35, 2.8)
(2, "Kiprun Trail Jaune taille 40", 50.00, 29.9, 20, 60, 20.10, 'KALENJI', 'Trail', "Chaussure de Trail Homme", 'libre', NULL, 0.390, 7.5, 9.8, 26.5)
(3, "Kiprun Trail Grise taille 39", 50, 28.55, 20, 60, 21.45, 'KALENJI', 'Trail', "Chaussure de Trail Femme", 'libre', NULL, 0.38, 7.5, 9.8, 25.5)
(4, "Survetement 560 Gym Bleu 10 ans", 4.8, 3.5, 20, 6, 1.3, 'DOMYOS', 'Gym', "Survetement Fille", 'libre', NULL, 0.49, 56, 36, 15)

**Stock :**
(1, 1, 4, 1, 15, 40, 55)
(2, 1, 4, 1, 6, 50, 56)
(3, 1, 4, 1, 4, 30, 34)
(1, 2, 5, 2, 8, 52, 60)
(3, 2, 5, 2, 8, 30, 38)
(4, 2, 5, 2, 26, 150, 176)

(2, 3, 6, 3, 2, 18, 20)
(3, 3, 6, 3, 8, 36, 44)
(4, 3, 6, 3, 15, 0, 15)

### C. Estimation de l'occupation en espace mémoire de l'entrepôt

Les estimations que nous allons donner concernent la taille de l'Entrepôt de Données après 12 mois d'utilisation pour chacun des Data-Mart que nous avons conçu. Afin de nous aider, nous avons utilisé des chiffres en provenance de diverses sources concernant l'enseigne "Décathlon".

#### Data-Mart "Vente"

En Juillet 2018, il existait 1412 **magasins** de l'enseigne "Décathlon" à travers le monde, et disposait de plus de 60000 références de **produits**. Ces chiffres ont été récupéré sur le site officiel de recrutement de l'enseigne Décathlon. On suppose que chaque magasin accueille en moyenne 15 600 **clients** par jour et que chaque client achète en moyenne 3 articles pour un panier moyen de 29,30 euros.

Une année civile est découpée en 365 jours ou 366 jours pour les années bissextiles. Les dates et horaires d'ouverture possibles sont différentes d'une culture à une autre (ouverture le dimanche, jours fériés différents, etc.). Afin de maximiser les résultats, nous supposerons que tous les magasins sont ouverts 7/7j.

Voici donc la taille estimée de chaque dimension :

- **Magasin** : 1 412 lignes d'après notre source précédente ;
- **Lieu** : 1412 lignes car chaque magasin est situé en un unique lieu ;
- **Client** : Sachant que chaque magasin accueille 15 600 clients par jour : 15600 * 1412 = 22 027 200 lignes par jour, et porté sur 366 jours, cela donne 8 039 928 000 lignes en supposant que tous les clients soient anonymes et que certains reviennent plusieurs fois dans l'année, expliquant ainsi que cela dépasse la population mondiale ;
- **Produit** : 60 000 lignes au moins, s'il on suit le nombre de référence estimé que possède Décathlon ;
- **Temps** : 86 400 lignes pour le nombre de secondes existants sur une journée de 24 heures ;
- **CurrentDate** : 10 248 lignes afin de représenter toutes les combinaisons jours et dates possibles qui reviennent de façon cyclique, dépendant des années bissextiles. Nous considérons des cycles maximums de 28 ans, donc 366 * 28 = 10 248 lignes.

Ainsi, pour 15 600 clients par jour réalisant l'achat de 3 articles en moyenne par passage en caisse dans chaque magasin : 15600 * 3 * 1412 = 66 081 600 lignes.

Donc, sur une année, que nous supposerons bissextile afin de maximiser le calcul : 88 108 800 * 366 = 24 185 865 600 lignes.

Nous avons alors la génération de **24 185 865 600 lignes** dans la table des faits "Vente" sur une année.

Au vu du nombre de lignes, nous pouvons suggérer de réaliser un partitionnement par zone géographique (Europe de l'Ouest, Europe de l'Est, Asie Sud-Est, Amérique du Sud, etc).

### Data-Mart "Stock"

Nous allons estimer que dans le pire des cas, chacun des 1412 **magasins** dispose de chaque référence **produit** vendue par l'enseigne Décathlon, quelque soit la quantité. Les snapshots sont réalisés quotidiennement.

Voici donc la taille estimée de chaque dimension :

- **Magasin** : 1 412 lignes d'après notre source précédente ;
- **Lieu** : 1412 lignes car chaque magasin est situé en un unique lieu ;
- **Produit** : 60 000 lignes au moins, s'il on suit le nombre de référence estimé que possède Décathlon ;
- **CurrentDate** : 10 248 lignes comme expliqué précédemment dans le Data-Mart "Vente".

Ainsi pour chaque magasin et pour chaque référence : 1412 * 1 * 60 000 = 84 720 000 lignes.

Et comme les magasins sont ouverts 7/7j : 84 720 00 * 366 = 31 007 520 000 lignes.

Nous avons donc la génération de **31 007 520 000 lignes** dans la table des faits "Stock" sur une année.

De même, nous suggérons un partitionnement de la table des faits par zone géographique.

## D. Estimation de l'espace mémoire nécessaire à l'entrepôt de données

Afin de calculer approximativement la taille en octet de chaque table sur 12 mois d'utilisation, nous avons dû définir la taille de chaque type utilisé :

- **Date** : fixé à 7 octets ;
- **Numeric** : de 1 à 22 octets en fonction de la précision et du décalage après la virgule. De façon empirique, nous avons déterminé la formule suivante afin d'approximer la taille d'un champs : $(taille/2) + 1$ ;
- **Varchar2** : en considérant qu'un caractère est codé sur un octet, la taille maximale en mode Standard (si le système est configuré sur MAX_STRING_SIZE = STANDARD) est de 4000 octets.

Nous nous sommes basé sur la documentation officielle d'Oracle. Nous pouvons donc déduire de la taille d'une ligne et donc du tableau si on reprend les estimations précédentes. Les types et précisions des champs de chaque tables sont spécifiées en annexe.

- **Dimension Magasin** : 167 octets par ligne, donc 167 octets * 1 412 lignes = 235 804 octets ;
- **Dimension Lieu** : 525 octets par ligne, donc 525 * 1412 lignes = 741 300 octets ;
- **Dimension Client** : 460 octets par ligne donc 460 octets * 8 039 928 000 lignes = 3,698 To environs ;
- **Dimension Produit** : 528 octets par ligne, donc 528 octets * 60 000 lignes = 31,68 Mo ;
- **Dimension Temps** : 48 octets par ligne, donc 48 octets * 86 400 lignes = 4,15 Mo environs ;
- **Dimension CurrentDate** :30 octets, donc 30 octets * 10 248 lignes = 307 440 octets ;
- **Table des faits Vente** : chaque ligne représente 45 octets en mémoire, donc 45 * 24 185 865 600 lignes = 1,088 To environs
- **Table des faits Stock** : chaque ligne représente 35 octets, donc 35 * 31 007 520 000 lignes = 1,085 To environs

Soit environs 5,871 To + K octets sachant K une constante comprenant le stockage des noms des colonnes et des tables, des méta-données éventuelles ainsi que la place en mémoire occupée par l'indexation des informations. Chaque SGBD dispose de sa propre façon de créer et de stocker les indexations.

# IV. Implémentation de la solution

## A. Implémentation sous Oracle et vues virtuelles

L'implémentation est en annexe dans les listings 2.1 à 2.4.

Une vue est un ensemble de relations déduites d'une base de données par compositions des relations entre les tables. Elles sont ici virtuelles car elle ne sont pas stockées dans la base de données. Les vues existent donc à des fins d'exploitation visuelle seulement.

```
-- Tables virtuelles liees a Vente

CREATE OR REPLACE VIEW VENTE_PRODUIT
AS SELECT * FROM Produit;

CREATE OR REPLACE VIEW VENTE_LIEU
AS SELECT * FROM Lieu;

CREATE OR REPLACE VIEW VENTE_DATE
AS SELECT * FROM CurrentDate;

CREATE OR REPLACE VIEW VENTE_MAGASIN
AS SELECT * FROM Magasin;

-- Tables virtuelles liees a Stock

CREATE OR REPLACE VIEW STOCK_PRODUIT
AS SELECT * FROM Produit;

CREATE OR REPLACE VIEW STOCK_LIEU
AS SELECT * FROM Lieu;

CREATE OR REPLACE VIEW STOCK_DATE
AS SELECT * FROM CurrentDate;

CREATE OR REPLACE VIEW STOCK_MAGASIN
AS SELECT * FROM Magasin;
```

Listing 1.1 – Vues virtuelles des dimensions partagées

## B. Traitement de requêtes analytiques

- Chiffre d'affaire total en fonction de la saison par magasin ?

```
SELECT cd.saison, m.idMagasin, AVG(v.prixVente) AS CA
FROM Vente v, VENTE_MAGASIN m, VENTE_DATE cd
WHERE v.idDate = cd.idDate
AND v.idMagasin = m.idMagasin
GROUP BY(cd.saison, m.idMagasin);
```

Listing 1.2 – Requête analytique n°1

- Quel est la quantité de produit vendu par catégorie et par magasin ?

```
SELECT m.idMagasin, p.categorie, v.quantiteVendue AS "Quantite"
FROM Vente v, VENTE_MAGASIN m, VENTE_PRODUIT p
WHERE v.idProduit = p.idProduit
AND v.idMagasin = m.idMagasin
GROUP BY (m.idMagasin, p.categorie, v.quantiteVendue)
ORDER BY m.idMagasin;
```

Listing 1.3 – Requête analytique n°2

- Quel est le total des ventes par magasin et par type de produit ?

```
SELECT m.idMagasin, p.categorie, SUM(v.prixVente) AS "Quantite"
FROM Vente v, VENTE_MAGASIN m, VENTE_PRODUIT p
WHERE v.idProduit = p.idProduit
AND v.idMagasin = m.idMagasin
GROUP BY (m.idMagasin, p.categorie, v.prixVente)
ORDER BY m.idMagasin, p.categorie;
```

Listing 1.4 – Requête analytique n°3

- Quel est le moment de la journée où il y a le plus de vente ?

```sql
SELECT t.indicateurAMPM, COUNT(v.prixVente)
FROM Vente v, Temps t
WHERE v.idTemps = t.idTemps
GROUP BY t.indicateurAMPM
ORDER BY COUNT(v.prixVente);
```

Listing 1.5 – Requête analytique n°4

- Quels sont les magasins ayant effectué le plus de vente en 2018 ?

```sql
SELECT COUNT(p.idProduit) , v.idMagasin
FROM VENTE_PRODUIT p, Vente v, VENTE_DATE d
WHERE p.idProduit = v.idProduit
AND v.idDate = d.idDate
AND d.annee = 2018
GROUP BY v.idMagasin
ORDER BY COUNT(p.idProduit) DESC;
```

Listing 1.6 – Requête analytique n°5

- Quelle est la quantité restante pour chaque produit par magasin ?

```sql
SELECT m.idMagasin, p.description, SUM(s.quantiteTotale)
FROM STOCK_MAGASIN m, STOCK_PRODUIT p, Stock s
WHERE m.idMagasin = s.idMagasin
AND p.idProduit = s.idProduit
GROUP BY m.idMagasin, p.description;
```

Listing 1.7 – Requête analytique n°6

- Quels sont les produits dont la quantité est inférieure à un certain seuil ?

```sql
SELECT m.idMagasin, p.idProduit, p.description, s.quantiteEntrepot
FROM STOCK_PRODUIT p, STOCK_MAGASIN m , Stock s
WHERE p.idProduit = s.idProduit
AND m.idMagasin = s.idMagasin
AND s.quantiteEntrepot < 50
GROUP BY m.idMagasin, p.idProduit, p.description, s.quantiteEntrepot;
```

Listing 1.8 – Requête analytique n°7

- Quels sont les produits dont la quantité est supérieure à un certain seuil ?

```sql
SELECT m.idMagasin, p.idProduit, p.description, s.quantiteEntrepot
FROM STOCK_PRODUIT p, STOCK_MAGASIN m , Stock s
WHERE p.idProduit = s.idProduit
AND m.idMagasin = s.idMagasin
AND s.quantiteEntrepot > 50
GROUP BY m.idMagasin, p.idProduit, p.description, s.quantiteEntrepot;
```

Listing 1.9 – Requête analytique n°8

- Quels sont les magasins ayant le plus de produits en stock à un jour donné ?

```sql

SELECT m.nomMagasin, SUM(s.quantiteTotale)
FROM STOCK_MAGASIN m, STOCK_DATE d, Stock s
WHERE m.idMagasin = s.idMagasin
AND d.idDate = s.idDate
AND d.jour <= 5
AND d.mois <= 11
AND d.annee <= 2018
GROUP BY m.nomMagasin;
```

Listing 1.10 – Requête analytique n°9

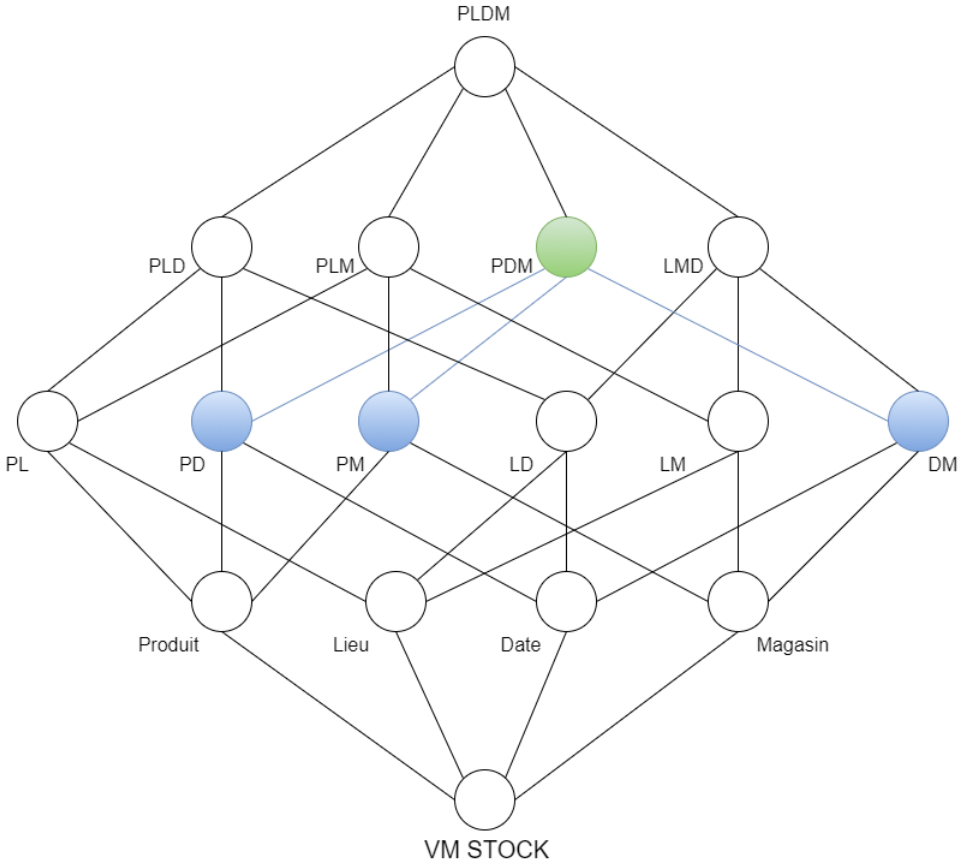
- Quels sont les produits prenant le plus de place en entrepôt par magasin ?

```sql
SELECT  m.idMagasin, p.idProduit, p.description, ((p.hauteur * p.largeur * p.
    profondeur) * s.quantiteEntrepot) / 1000000 AS "Volume"
FROM STOCK_PRODUIT p, STOCK_MAGASIN m, Stock s
WHERE m.idMagasin = s.idMagasin
AND p.idProduit = s.idProduit
GROUP BY m.idMagasin, p.idProduit, p.description, p.hauteur,  p.largeur,  p.
    profondeur,  s.quantiteEntrepot
ORDER BY ((p.hauteur * p.largeur * p.profondeur) * s.quantiteEntrepot) DESC;
```

Listing 1.11 – Requête analytique n°10

## C. Vues matérialisées permettant de répondre aux requêtes analytiques

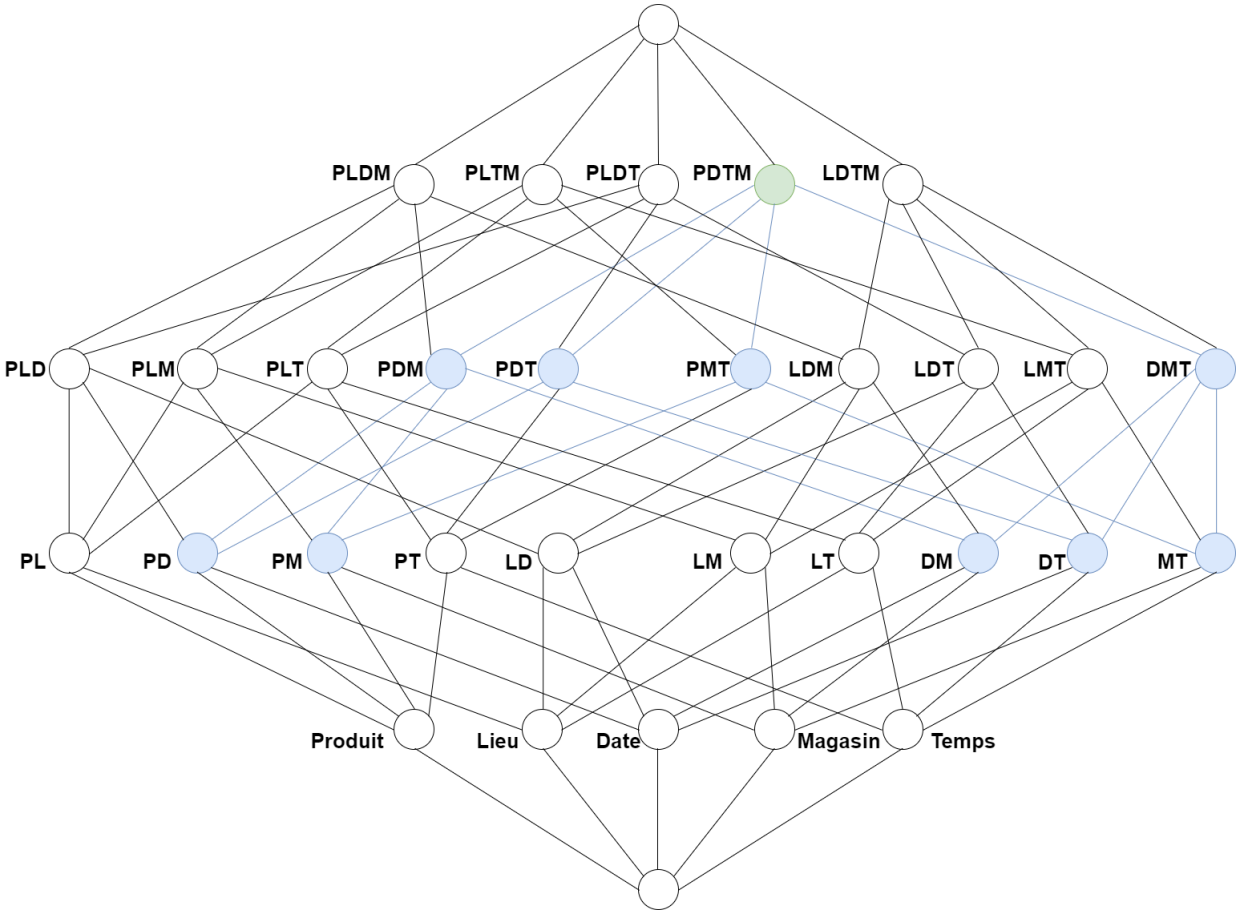
### Vue matérialisées issues du Data-Mart Stock




```sql
CREATE MATERIALIZED VIEW MV_SPDM
AS
  SELECT m.idMagasin, m.nomMagasin, p.idProduit, p.description, p.hauteur, p.largeur, p.
      profondeur, s.quantiteEntrepot, s.quantiteTotale, d.idDate, d.jour, d.mois, d.annee
  FROM Magasin m, Produit p, CurrentDate d, Stock s
  WHERE s.idMagasin = m.idMagasin
  AND s.idProduit = p.idProduit
  AND s.idDate = d.idDate;
```

Listing 1.12 – Vue matérialisée pour les requêtes sur la table des faits Stock

### Vue matérialisées issues du Data-Mart Vente




```sql
CREATE MATERIALIZED VIEW MV_VPDMT
AS
  SELECT m.idMagasin, p.catergorie, t.idTemps, t.indicateurAMPM, d.idDate, d.annee, d.saison,
      v.prixVente, v.quantiteVendue
  FROM Produit p, CurrentDate d, Magasin m, Temps t
  WHERE v.idMagasin = m.idMagasin
  AND v.idProduit = p.idProduit
  AND v.idDate = d.idDate
  AND v.idTemps = t.idTemps;
```

Listing 1.13 – Vue matérialisée pour les requêtes sur la table des faits Stock

L'utilisation des vues matérialisées par rapport aux vues virtuelles comporte de nombreux avantages. Elles nous offrent la possibilité de stocker des résultats de requêtes qui peuvent être lourdes et/ou complexes, faire des réplications de tables ainsi que des pré-agrégations accessibles facilement et rapidement pour d'autres requêtes. Son utilité première est donc d'optimiser les performances, chose essentielle dans un Entrepôt de Données.

Celles-ci ne sont d'ailleurs pas recalculées à chaque appel, elle peuvent néanmoins subir un rafraîchissement en fonction des besoins (ON COMMIT par exemple). Ainsi, le SGBD peut tirer pleinement partie de leur création en les utilisant lorsque c'est nécessaire et jugé plus rapide par l'optimiseur.

# 2 Annexes

## I. Implémentation sous Oracle

```
DROP TABLE Vente;
DROP TABLE Stock;
DROP TABLE Client;
DROP TABLE Produit;
DROP TABLE Temps;
DROP TABLE Lieu;
DROP TABLE CurrentDate;
DROP TABLE Magasin;

CREATE TABLE Magasin(
  idMagasin          NUMERIC(8) NOT NULL,
  nomMagasin         VARCHAR2(64) NOT NULL,
  numeroMagasin        NUMERIC(6) NOT NULL,
  nbEmployees        NUMERIC(6),
  gerantMagasin      VARCHAR2(64),
  SurfaceDuMagasin     NUMERIC(6),
  SurfaceDeStockage    NUMERIC(6),
  SurfaceDeVente       NUMERIC(6),
  datePremiereOuverture DATE NOT NULL,
  dateFermetureMagasin  DATE
);

CREATE TABLE CurrentDate(
  idDate           NUMERIC(8) NOT NULL,
  jour           NUMERIC(2) NOT NULL,
  jourSemaine        NUMERIC(1) NOT NULL,
  jourAnnee        NUMERIC(3) NOT NULL,
  semaine          NUMERIC(2) NOT NULL,
  semaineDuMois        NUMERIC(1) NOT NULL,
  mois           NUMERIC(2) NOT NULL,
  trimestre          NUMERIC(1) NOT NULL,
  annee            NUMERIC(4) NOT NULL,
  saison           VARCHAR2(12) NOT NULL,
  vacances         NUMERIC(1)
);

CREATE TABLE Lieu(
  idLieu           NUMERIC(8) NOT NULL,
  pays           VARCHAR2(64) NOT NULL,
  departement        VARCHAR2(64) NOT NULL,
  region           VARCHAR2(64) NOT NULL,
  ville            VARCHAR2(128) NOT NULL,
  quartier         VARCHAR2(64),
  rue            VARCHAR2(128),
  codeCommune        NUMERIC(5),
  zipCode          NUMERIC(5) NOT NULL,
  cedex          NUMERIC(2)
);

CREATE TABLE Temps(
  idTemps          NUMERIC(8) NOT NULL,
  heure          NUMERIC(2) NOT NULL,
  minute           NUMERIC(2) NOT NULL,
  seconde          NUMERIC(2) NOT NULL,
  fuseauHoraire        VARCHAR2(3) NOT NULL,
  indicateurAMPM       VARCHAR2(2) NOT NULL,
  momentJournee      VARCHAR2(32)
);

```

```sql
CREATE TABLE Produit(
  idProduit       NUMERIC(8) NOT NULL,
  description       VARCHAR2(128) NOT NULL,
  prixHT          NUMERIC(8,2) NOT NULL,
  prixAchatFournisseur  NUMERIC(8,2) NOT NULL,
  tauxTaxe        NUMERIC(4,2) NOT NULL,
  prixTTC         NUMERIC(8,2) NOT NULL,
  marge           NUMERIC(8,2) NOT NULL,
  marque          VARCHAR2(64) NOT NULL,
  categorie         VARCHAR2(128) NOT NULL,
  sousCategorie       VARCHAR2(128) NOT NULL,
  typeConditionnement   VARCHAR2(32),
  tailleConditionnement   NUMERIC(3),
  poidsNet        NUMERIC(5,2),
  hauteur         NUMERIC(4),
  largeur         NUMERIC(4),
  profondeur        NUMERIC(4)
);

CREATE TABLE Client(
  idClient        NUMERIC(8) NOT NULL,
  prenom          VARCHAR2(64),
  nom           VARCHAR2(64),
  adresse         VARCHAR2(64),
  ville           VARCHAR2(128),
  mail          VARCHAR2(128),
  dateProgrammeFidelite DATE
);

CREATE TABLE Stock(
  idProduit         NUMERIC(8) NOT NULL,
  idLieu          NUMERIC(8) NOT NULL,
  idDate          NUMERIC(8) NOT NULL,
  idMagasin         NUMERIC(8) NOT NULL,
  quantiteEnRayon     NUMERIC(8) NOT NULL,
  quantiteEntrepot    NUMERIC(8) NOT NULL,
  quantiteTotale      NUMERIC(8) NOT NULL
);

CREATE TABLE Vente(
  idClient        NUMERIC(8) NOT NULL,
  idProduit       NUMERIC(8) NOT NULL,
  idTemps         NUMERIC(8) NOT NULL,
  idLieu          NUMERIC(8) NOT NULL,
  idDate          NUMERIC(8) NOT NULL,
  idMagasin         NUMERIC(8) NOT NULL,
  numeroTransaction     NUMERIC(8) NOT NULL,
  quantiteVendue      NUMERIC(8) NOT NULL,
  prixVente         NUMERIC(8) NOT NULL
);
```

Listing 2.1 – Création des tables

```sql
ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_jour CHECK (jour BETWEEN 0 AND 31);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_jourSemaine CHECK (jourSemaine BETWEEN 1 AND 7);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_jourAnnee CHECK (jourAnnee BETWEEN 1 AND 366);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_semaine CHECK (semaine BETWEEN 1 AND 53);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_semaineDuMois CHECK (semaineDuMois BETWEEN 1 AND
    5);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_mois CHECK (mois BETWEEN 1 AND 12);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_trimestre CHECK (trimestre BETWEEN 1 AND 4);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_annee CHECK (annee >= 1976);

ALTER TABLE CurrentDate ADD CONSTRAINT CHECK_vacances CHECK (vacances BETWEEN 0 AND 1);

ALTER TABLE Temps ADD CONSTRAINT CHECK_heure CHECK (heure BETWEEN 0 AND 12);

ALTER TABLE Temps ADD CONSTRAINT CHECK_minute CHECK (minute BETWEEN 0 AND 60);

```

```sql
ALTER TABLE Temps ADD CONSTRAINT CHECK_seconde CHECK (seconde BETWEEN 0 AND 60);

ALTER TABLE Temps ADD CONSTRAINT CHECK_indicateurAMPM CHECK (indicateurAMPM IN ('AM', 'PM'));

ALTER TABLE Temps ADD CONSTRAINT CHECK_momentJourneee CHECK (momentJournee IN ('Matin', 'Midi'
    , 'Apres-Midi', 'Soir'));

ALTER TABLE Produit ADD CONSTRAINT CHECK_prixHT CHECK (prixHT >= 0.0);

ALTER TABLE Produit ADD CONSTRAINT CHECK_prixAchatFournisseur CHECK (prixAchatFournisseur >=
    0.0);

ALTER TABLE Produit ADD CONSTRAINT CHECK_tauxTaxe CHECK (tauxTaxe BETWEEN 0 AND 99);

ALTER TABLE Produit ADD CONSTRAINT CHECK_prixTTC CHECK (prixTTC >= 0.0);

ALTER TABLE Produit ADD CONSTRAINT CHECK_marge CHECK (marge >= 0.0);

ALTER TABLE Stock ADD CONSTRAINT CHECK_quantiteEnRayon CHECK (quantiteEnRayon >= 0);

ALTER TABLE Stock ADD CONSTRAINT CHECK_quantiteEntrepot CHECK (quantiteEntrepot >= 0);

ALTER TABLE Stock ADD CONSTRAINT CHECK_quantiteTotale CHECK (quantiteTotale >= 0);

ALTER TABLE Vente ADD CONSTRAINT CHECK_prixVente CHECK (prixVente >= 0.0);
```

Listing 2.2 – Ajout des contraintes d'intégrité

```sql
-- Magasin - clear

INSERT INTO Magasin (idMagasin, nomMagasin, numeroMagasin, nbEmployees, gerantMagasin,
    SurfaceDuMagasin, SurfaceDeStockage, SurfaceDeVente, datePremiereOuverture,
    dateFermetureMagasin) VALUES (1, 'Montpellier-Odysseum', 157, 199, 'Jean-Michel Amoitier',
     3900, 1000, 2600, TO_DATE('2008/09/01', 'YYYY/MM/DD'), NULL);

INSERT INTO Magasin (idMagasin, nomMagasin, numeroMagasin, nbEmployees, gerantMagasin,
    SurfaceDuMagasin, SurfaceDeStockage, SurfaceDeVente, datePremiereOuverture,
    dateFermetureMagasin) VALUES (2, 'Quetigny-Dijon', 82, 186, 'Jean-Michel Apeupres', 3500,
    1000, 2200, TO_DATE('2009/02/11', 'YYYY/MM/DD'), NULL);

INSERT INTO Magasin (idMagasin, nomMagasin, numeroMagasin, nbEmployees, gerantMagasin,
    SurfaceDuMagasin, SurfaceDeStockage, SurfaceDeVente, datePremiereOuverture,
    dateFermetureMagasin) VALUES (3, 'Orange', 329, 198, 'Jean-Michel Padevoix', 3900, 1200,
    2600, TO_DATE('2007/09/01', 'YYYY/MM/DD'), NULL);


-- CurrentDate - clear

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (1, 29, 1, 302, 43, 5, 10, 4, 2018, 'Automne',
    1);

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (2, 30, 2, 303, 43, 5, 10, 4, 2018, 'Automne',
    1);

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (3, 31, 3, 304, 43, 5, 10, 4, 2018, 'Automne',
    1);

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (4, 1, 4, 305, 43, 1, 11, 4, 2018, 'Automne',
    1);

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (5, 2, 5, 306, 43, 1, 11, 4, 2018, 'Automne',
    1);

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (6, 3, 6, 307, 43, 1, 11, 4, 2018, 'Automne',
    1);

INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
    trimestre, annee, saison, vacances) VALUES (7, 14, 4, 73, 11, 3, 3, 1, 2019, 'Printemps',
    0);

```

```
26 INSERT INTO CurrentDate (idDate, jour, jourSemaine, jourAnnee, semaine, semaineDuMois, mois,
       trimestre, annee, saison, vacances) VALUES (8, 15, 5, 74, 11, 3, 3, 1, 2019, 'Printemps',
       0);
27
28
29 -- Lieu - clear
30
31 INSERT INTO Lieu (idLieu, pays, departement, region, ville, quartier, rue, codeCommune,
       zipCode, cedex) VALUES (1, 'France', 'Herault', 'Occitanie', 'Montpellier', 'Millenaire',
       '1072 Avenue Georges Melies', 34172, 34000, 4);
32
33 INSERT INTO Lieu (idLieu, pays, departement, region, ville, quartier, rue, codeCommune,
       zipCode, cedex) VALUES (2, 'France', 'Cote d-Or', 'Bourgogne-Franche-Comte', 'Quetigny', '
       ZI Quetigny', 'Rue Champeau', 21515, 21800, NULL);
34
35 INSERT INTO Lieu (idLieu, pays, departement, region, ville, quartier, rue, codeCommune,
       zipCode, cedex) VALUES (3, 'France', 'Vaucluse', 'Provence-Alpes-Cote-D-Azur', 'Orange', '
       ZA Porte Sud', 'ZA Porte Sud', 84087, 84100, NULL);
36
37
38 -- Temps - clear
39
40 INSERT INTO Temps (idTemps, heure, minute, seconde, fuseauHoraire, indicateurAMPM,
       momentJournee) VALUES (1, 2, 40, 29, '+01', 'PM', 'Apres-Midi');
41
42 INSERT INTO Temps (idTemps, heure, minute, seconde, fuseauHoraire, indicateurAMPM,
       momentJournee) VALUES (2, 3, 16, 47, '+01', 'PM', 'Apres-Midi');
43
44 INSERT INTO Temps (idTemps, heure, minute, seconde, fuseauHoraire, indicateurAMPM,
       momentJournee) VALUES (3, 11, 48, 56, '+01', 'AM', 'Matin');
45
46 INSERT INTO Temps (idTemps, heure, minute, seconde, fuseauHoraire, indicateurAMPM,
       momentJournee) VALUES (4, 3, 33, 9, '+01', 'PM', 'Apres-Midi');
47
48
49 -- Produit - clear
50
51 INSERT INTO Produit (idProduit, description, prixHT, prixAchatFournisseur, tauxTaxe, prixTTC,
       marge, marque, categorie, sousCategorie, typeConditionnement, tailleConditionnement,
       poidsNet, hauteur, largeur, profondeur) VALUES (1, 'Expert TR 960', 80.00, 55.59, 20, 96,
       24.41, 'ARTENGO', 'Tennis', 'Raquette Tennis Adulte', 'libre', NULL, 0.360, 86.5, 35, 2.8)
       ;
52
53 INSERT INTO Produit (idProduit, description, prixHT, prixAchatFournisseur, tauxTaxe, prixTTC,
       marge, marque, categorie, sousCategorie, typeConditionnement, tailleConditionnement,
       poidsNet, hauteur, largeur, profondeur) VALUES (2, 'Kiprun Trail Jaune taille 40', 50.00,
       29.9, 20, 60, 20.10, 'KALENJI', 'Trail', 'Chaussure de Trail Homme', 'libre', NULL, 0.390,
        7.5, 9.8, 26.5);
54
55 INSERT INTO Produit (idProduit, description, prixHT, prixAchatFournisseur, tauxTaxe, prixTTC,
       marge, marque, categorie, sousCategorie, typeConditionnement, tailleConditionnement,
       poidsNet, hauteur, largeur, profondeur) VALUES (3, 'Kiprun Trail Grise taille 39', 50,
       28.55, 20, 60, 21.45, 'KALENJI', 'Trail', 'Chaussure de Trail Femme', 'libre', NULL, 0.38,
        7.5, 9.8, 25.5);
56
57 INSERT INTO Produit (idProduit, description, prixHT, prixAchatFournisseur, tauxTaxe, prixTTC,
       marge, marque, categorie, sousCategorie, typeConditionnement, tailleConditionnement,
       poidsNet, hauteur, largeur, profondeur) VALUES (4, 'Survetement 560 Gym Bleu 10 ans', 4.8,
        3.5, 20, 6, 1.3, 'DOMYOS', 'Gym', 'Survetement Fille', 'libre', NULL, 0.49, 56, 36, 15);
58
59
60 -- Client - clear
61
62 INSERT INTO Client (idClient, prenom, nom, adresse, ville, mail, dateProgrammeFidelite) VALUES
        (1, 'Michel', 'Bergeait', '55 Rue du Parvis', 'Grabels', 'michel.berger@mail.fr', '
       2009/08/11');
63
64 INSERT INTO Client (idClient, prenom, nom, adresse, ville, mail, dateProgrammeFidelite) VALUES
        (2, NULL, NULL, NULL, 'Quetigny', 'david.goodenough@mail.fr', NULL);
65
66 INSERT INTO Client (idClient, prenom, nom, adresse, ville, mail, dateProgrammeFidelite) VALUES
        (3, 'Henri', 'Lechanteur', '256 rue des oiseaux', 'Orange', 'henri.cestmoi@mail.fr', '
       2014/03/15');
67
```

```sql
INSERT INTO Client (idClient, prenom, nom, adresse, ville, mail, dateProgrammeFidelite) VALUES
    (4, 'David', 'Goodenought', '40 quai des Carrosses', 'Montpellier', 'matuture34@mail.fr',
    '2018/10/30');


-- Stock - clear

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (1, 1, 4, 1, 15, 40, 55);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (2, 1, 4, 1, 6, 50, 56);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (3, 1, 4, 1, 4, 30, 34);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (1, 2, 5, 2, 8, 52, 60);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (3, 2, 5, 2, 8, 30, 38);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (4, 2, 5, 2, 26, 150, 176);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (2, 3, 6, 3, 2, 18, 20);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (3, 3, 6, 3, 8, 36, 44);

INSERT INTO Stock (idProduit, idLieu, idDate, idMagasin, quantiteEnRayon, quantiteEntrepot,
    quantiteTotale) VALUES (4, 3, 6, 3, 15, 0, 15);


-- Vente - clear

INSERT INTO Vente (idClient, idProduit, idTemps, idLieu, idDate, idMagasin, numeroTransaction,
    quantiteVendue, prixVente) VALUES (1, 1, 1, 1, 1, 1, 2, 2, 184.00);

INSERT INTO Vente (idClient, idProduit, idTemps, idLieu, idDate, idMagasin, numeroTransaction,
    quantiteVendue, prixVente) VALUES (2, 2, 2, 2, 1, 2, 2, 1, 60.00);

INSERT INTO Vente (idClient, idProduit, idTemps, idLieu, idDate, idMagasin, numeroTransaction,
    quantiteVendue, prixVente) VALUES (3, 3, 3, 3, 2, 3, 3, 1, 60.00);

INSERT INTO Vente (idClient, idProduit, idTemps, idLieu, idDate, idMagasin, numeroTransaction,
    quantiteVendue, prixVente) VALUES (4, 4, 4, 1, 3, 1, 4, 3, 18.00);

INSERT INTO Vente (idClient, idProduit, idTemps, idLieu, idDate, idMagasin, numeroTransaction,
    quantiteVendue, prixVente) VALUES (1, 2, 2, 1, 7, 1, 5, 1, 60.00);

INSERT INTO Vente (idClient, idProduit, idTemps, idLieu, idDate, idMagasin, numeroTransaction,
    quantiteVendue, prixVente) VALUES (2, 1, 3, 2, 8, 2, 6, 1, 92.00);
```

Listing 2.3 – Insertion des données

```sql
-- Magasin
ALTER TABLE Magasin ADD PRIMARY KEY(idMagasin);

-- CurrentDate
ALTER TABLE CurrentDate ADD PRIMARY KEY(idDate);

-- Lieu
ALTER TABLE Lieu ADD PRIMARY KEY(idLieu);

-- Temps
ALTER TABLE Temps ADD PRIMARY KEY(idTemps);

-- Produit
ALTER TABLE Produit ADD PRIMARY KEY(idProduit);

-- Client
ALTER TABLE Client ADD PRIMARY KEY(idClient);

-- Stock
```

```
20 ALTER TABLE Stock ADD FOREIGN KEY(idMagasin) REFERENCES Magasin(idMagasin);
21 ALTER TABLE Stock ADD FOREIGN KEY(idDate) REFERENCES CurrentDate(idDate);
22 ALTER TABLE Stock ADD FOREIGN KEY(idLieu) REFERENCES Lieu(idLieu);
23 ALTER TABLE Stock ADD FOREIGN KEY(idProduit) REFERENCES Produit(idProduit);
24 ALTER TABLE Stock ADD PRIMARY KEY(idMagasin, idDate, idLieu, idProduit);
25
26 -- Vente
27 ALTER TABLE Vente ADD FOREIGN KEY(idMagasin) REFERENCES Magasin(idMagasin);
28 ALTER TABLE Vente ADD FOREIGN KEY(idDate) REFERENCES CurrentDate(idDate);
29 ALTER TABLE Vente ADD FOREIGN KEY(idLieu) REFERENCES Lieu(idLieu);
30 ALTER TABLE Vente ADD FOREIGN KEY(idTemps) REFERENCES Temps(idTemps);
31 ALTER TABLE Vente ADD FOREIGN KEY(idProduit) REFERENCES Produit(idProduit);
32 ALTER TABLE Vente ADD FOREIGN KEY(idClient) REFERENCES Client(idClient);
33 ALTER TABLE Vente ADD PRIMARY KEY(idMagasin, idDate, idLieu, idTemps, idProduit, idClient);
```

Listing 2.4 – Ajout des clefs primaires et étrangères