



Algorithme de matérialisation partielle – Projet ODE

|  |  |  |
| --- | --- | --- |
| **Suivi des versions** | | |
| **Date** | **Version** | **Objet de la version** |
| 02/08/2015 | 01 | [OLIVIER] Création du document |
|  |  |  |
|  |  |  |

Sommaire

[Analyse des dimensions du modèle de cube 3](#_Toc426298978)

[Reports d’utilisation 4](#_Toc426298979)

[Pour les métiers de la Direction 4](#_Toc426298980)

[Pour les métiers du Marketing 4](#_Toc426298981)

[Volumétrie de chaque attributs 5](#_Toc426298982)

[Algorithme de matérialisation partielle 6](#_Toc426298983)

[Entrées 6](#_Toc426298984)

[Traitements 6](#_Toc426298985)

[Volumétrie des combinaisons de dimensions 6](#_Toc426298986)

[Initialisation de l’algorithme 6](#_Toc426298987)

[Déroulé de l’algorithme 6](#_Toc426298988)

[Sorties 6](#_Toc426298989)

[Améliorations possibles 6](#_Toc426298990)

[Affinement de la volumétrie de chaque attributs 6](#_Toc426298991)

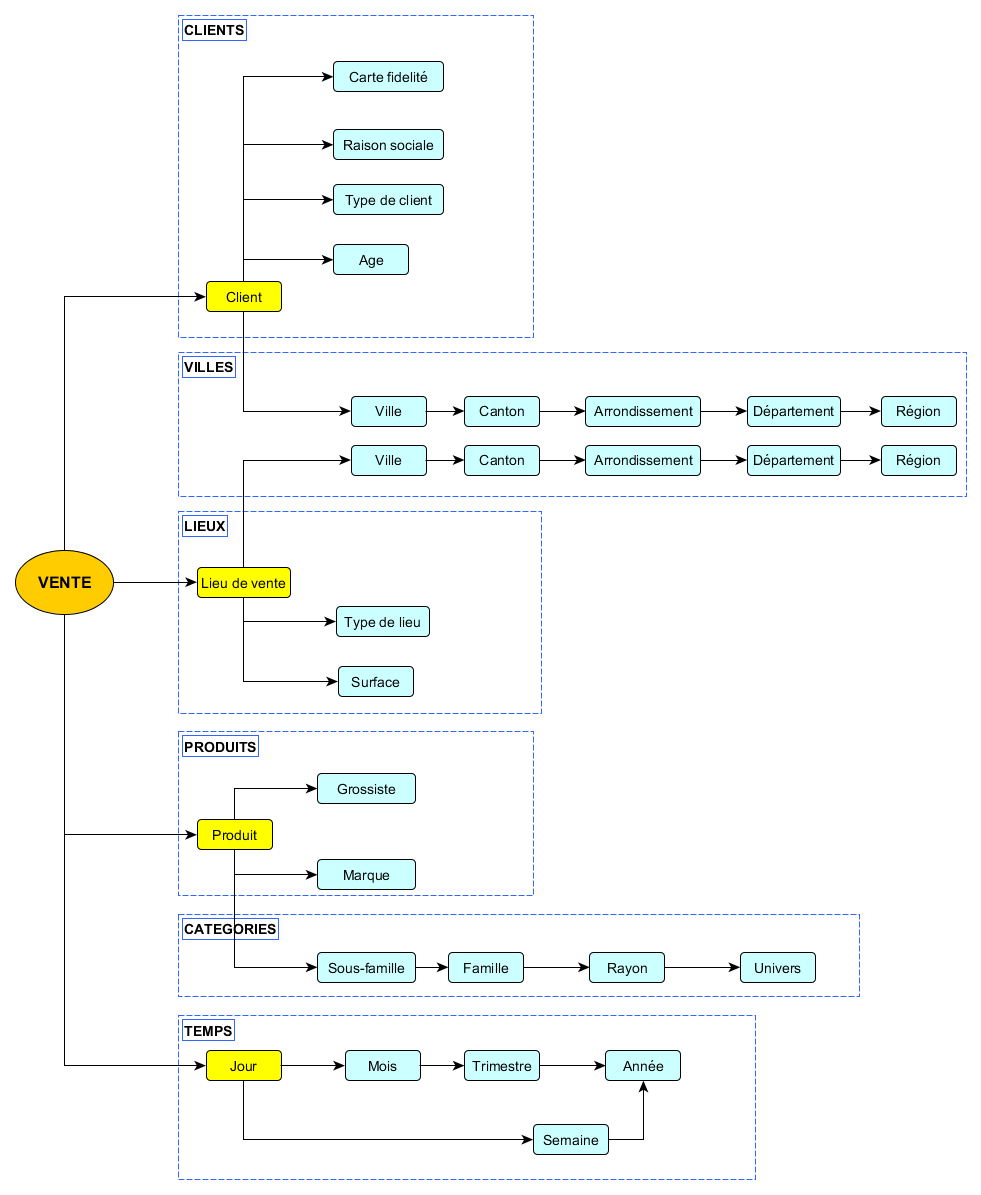
[Discrétisation de données continues 6](#_Toc426298992)

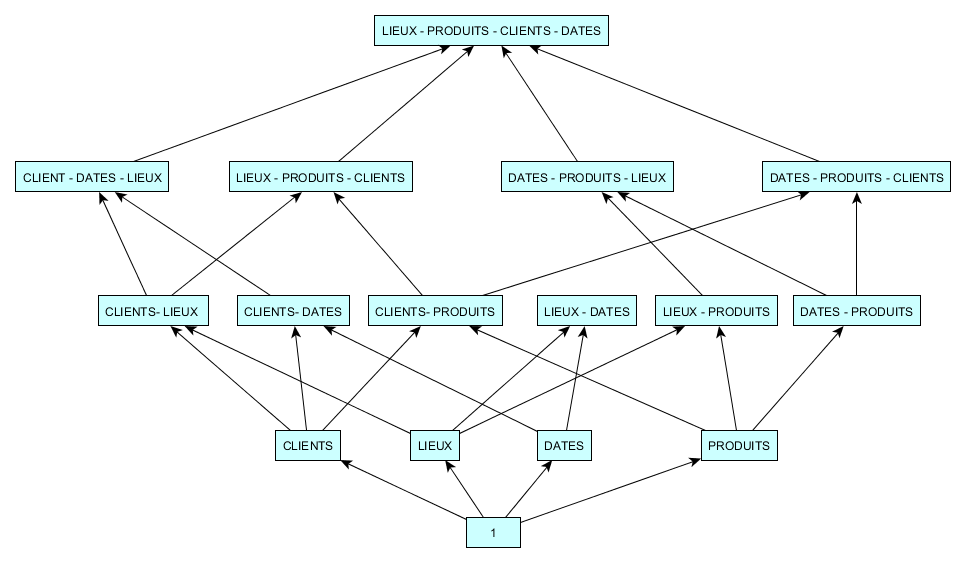
[Impact de la hiérarchie des dimensions 7](#_Toc426298993)

[Annexe 7](#_Toc426298994)

[Script de volumétrie de l’entrepôt de données 7](#_Toc426298995)

# Analyse des dimensions du modèle de cube





## Volumétrie de chaque attributs

* **Sur la dimension des clients**, il existe NC enregistrements.
* **Sur la dimension des lieux**, il existe NL enregistrements.
* **Sur la dimension des dates**, il existe ND enregistrements.
* **Sur la dimension des produits**, il existe NP enregistrements.

La modélisation en flocon va permettre de chiffrer ces volumes par de simples requêtes « Count(\*) » sur la table de faits des VENTES, groupés sur le ou les attributs du tuple.

On en tire les volumétries suivantes :

|  |  |  |  |
| --- | --- | --- | --- |
| Vue | Attributs du tuple | Volumétrie | Volumétrie réelle |
| V0 | ∅ | 1 | 1 |
| VC | Client | NC |  |
| VL | Lieu | NL |  |
| VD | Date | ND |  |
| VP | Produit | NP |  |
| VCL | Client – Lieu | NCL ≤ NC \* NL |  |
| VCD | Client – Date | NCD ≤ NC \* ND |  |
| VCP | Client – Produit | NCP ≤ NC \* NP |  |
| VLD | Lieu – Date | NLD ≤ NL \* ND |  |
| VLP | Lieu – Produit | NLP ≤ NL \* NP |  |
| VDP | Date – Produit | NDP ≤ ND \* NP |  |
| VCDL | Client – Date – Lieu | NCDL ≤ NC \* ND \* NL |  |
| VLPC | Lieu – Produit – Client | NLPC ≤ NL \* NP \* NC |  |
| VDPL | Date – Produit – Lieu | NDPL ≤ ND \* NP \* NL |  |
| VDPC | Date – Produit – Client | NDPC ≤ ND \* NP \* NC |  |
| VLPCD | Lieu – Produit – Client - Date | NLPCD ≤ NL \* NP \* NC \* ND |  |

Les signes d’infériorité rappellent qu’il s’agit de maximum. Pr exemple, le nombre de tuples « **Client – Produit** » est composé d’autant de lignes que de client ayant acheté ce produit au moins une fois. Comme tous les clients n’ont pas achetés tous les produits, on aura moins de **NC \* NP** tuples.

# Algorithme de matérialisation partielle

## Présentation

Dans le cours D111 – Informatique décisionnelle, Sofian MAABOUT présente un algorithme de matérialisation partielle du cube OLAP. Il s’agit d’un équilibre entre :

* **Aucune matérialisation** : L’espace disque du cube est réduite au seules données de l’entrepôt de données qui a servi de source. Les performances peuvent être décevantes du fait du nombre d’accès disque. Des index judicieusement placés et un sens de parcours adéquat permettent de limiter ces problèmes.
* **Matérialisation totale** : L’ensemble des calculs de sommes est réalisé à l’avance, et stocké sur le disque dur. L’interrogation du cube se limite alors à la seule lecture de données déjà calculés, et les performances sont optimales. Toutefois, l’espace disque limité ne permet pas toujours la mise en œuvre de cette logique

La matérialisation partielle consiste à choisir au mieux un sous-ensemble de données du cube à pré calculer, sous forme de vue matérialisés. Lorsque le moteur recevra une requête OLPA, il ira soit chercher dans ces vues matérialisées, soit fera le calcul à la volée.

Cet algorithme supporte une seule contrainte, par exemple le nombre maximum de vues à matérialiser. Ici, nous fixerons la contrainte sur l’espace disque maximum à utiliser par les vues matérialisés.

En théorie, à contrainte égale la solution de cet algorithme est supérieure ou égale à la solution optimale, obtenue par méthode analytique ou par essai de toutes les combinaisons possibles.

## Entrées

* Espace maximale de stockage sur disque, en Mo, sous forme d’entier
* Stockage fixe de la structure du cube (Chaines de dépendances des différentes vues possibles)

## Traitements

### Initialisation de l’algorithme

1. Nom, volumétrie et occupation disque totale des dimensions 1D, sous forme de série de tuples { ***nomDimension, comptageDimension, occupationDisque*** }

***Exemple****: La dimension « CLIENTS » = {« DIM\_CLIENTS », 12 345 lignes, 1 234 ko occupés par la table}*

1. Calcul des combinaisons de vues 2D, 3D et 4D des dimensions à partir des vues 1D
2. Génération et envoi des requêtes à SSAS pour obtenir les mêmes informations que les dimensions 1D
3. On construit une liste des vues sélectionnées « ***ListeVuesSelectionnees*** »  pour la matérialisation, contenant seulement VLPCD au départ.
4. Initialiser une liste de vues disponibles « ***ListeVuesDisponibles*** », contentant toutes les vues sauf VLPCD

### 

### Déroulé de l’algorithme

1. Remettre à zéro une variable temporaire « ***meilleureVue*** » associée à un bénéfice nul
2. Sur l’ensemble des vues encore présentes dans « ***ListeVuesDisponibles*** »:
3. D
4. D
5. Sommer les bénéfices unitaires comme la différence **AncienCout**(Vue en cours) et **NouveauCout**(Vue en cours)
6. Si le bénéfice de la vue en cours est supérieur à celui de « ***meilleureVue*** » : Sélectionner cette vue comme nouvelle « ***meilleureVue*** »
7. Quand toutes les vues de« ***ListeVuesDisponibles*** » ont été parcourues :
8. Mettre la vue sélectionnée « ***meilleureVue*** » dans « ***ListeVuesSelectionnees*** »
9. Supprimer « ***meilleureVue*** » de « ***ListeVuesDisponibles*** »
10. Soustraire l’occupation disque de la vue sélectionnée du total disponible restant.
11. Go To [1] s’il reste encore des vues dans « ***ListeVuesDisponibles*** » et qu’il reste de l’espace disque disponible et que l’itération précédente a bien sélectionné une vue (Pour éviter la boucle infinie si l’espace restant ne suffit à aucune vue restante)

## Sorties

* Modèle d’agrégats, qui sera appliqué sur SSAS. On matérialisera toutes les vues de la liste « ***ListeVuesSelectionnees*** »

# Implémentation technique

## Fonctionnalités requises

1. Se connecter à SSAS, en fournissant le nom de la base SSAS et du cube à traiter
2. Récupérer le nom de toutes les dimensions 1D du cube (Métadonnées)
3. Récupérer la volumétrie (Count) d’une table unique, groupée sur une ou plusieurs de ses colonnes (Requête MDX ?)
4. Récupérer l’occupation disque totale d’une dimension (Métadonnées)
5. Former une requête de schéma d’agrégation
6. Envoyer à SSAS la requête de schéma d’agrégation
7. Envoyer à SSAS la demande de recalcule des agrégats
8. Se déconnecter de SSAS

## Analysis Management Objects OLAP

On utilisera la librairie de classes **Analysis Management Objects (AMO)** pour implémenter les fonctionnalités de manipulation d’agrégats :

* \*\*\*\*\*\*

Un lien vers la section dédié MSDN « ***Multidimensional Model Programming*** » est donné en annexe.

## Librairie ADOMD.NET

On utilisera la librairie **ADOMD.NET** pour se connecter à SSAS et implémenter une partie des fonctionnalités requises :

* Exécution de XMLA « brut » via ***CreateCommand()***, ***CommandText*** puis ***ExecuteNonQuery()***
* Recherche de métadonnées du cube via \*\*\*\*\*\*\*

Un lien vers la section dédié MSDN « ***ADOMD.NET Client Programming*** » est donné en annexe.

## Conception en vue de passage en Web-service

Lors de la conception du code, on s’attachera à séparer les fonctions en 2 groupes :

* **Un groupe de fonctions dédiées aux interactions avec SSAS**, au travers de la librairie ADOMD.NET et/ou de code XMLA. Elles constitueront plus tard le client du Web-service de calcul.
* **Un groupe de fonctions dédiées aux calculs à proprement parler**, sans utilisation directe de la librairie ADOMD.NET. Elles constitueront plus tard le Web-service de calcul.

# Améliorations possibles

## Affinement de la volumétrie de chaque attributs

On conservera le niveau le plus bas de l’ensemble des reports.

* **Sur la dimension des clients**, le niveau le plus fin est celui de la ville des clients pour y récupérer la population (Reports R9 et R10)
* **Sur la dimension des lieux**, le niveau le plus fin est le magasin (Report R9)
* **Sur la dimension des dates**, le niveau le plus fin est le mois (Reports R1, R3, R5 à R10)
* **Sur la dimension des produits**, le niveau le plus fin est celui de la famille des produits (Reports R6 et R8)

D’où les quantifications suivantes :

* **Nombre d’items « Clients »**: Autant que de villes liées aux clients enregistrés, car une valeur de population par ville.
* **Nombre d’items « Lieux »** : Autant que de points de vente (Magasins et site Internet)
* **Nombre d’items « Dates »**: Autant que de nombre de couples {Mois ; Année}
* **Nombre d’items « Produits »**: Autant que de familles de produits

Cette approche va fortement diminuer la volumétrie de certaines dimensions, et permettra de stocker plus de vue matérialisées dans le même espace disque.

Toutefois, cette approche n’est pas adaptée pour l’utilisation de type « *Business Discovery* » (*QlikView*, *Excel* et *Power Pivot*…) car on ne connait pas à l’avance les requêtes MDX qui seront faites sur le cube, et donc on ne peut pas prévoir efficacement les « raccourcis » de la matérialisation partielle.

## Discrétisation de données continues

Les informations « continues » des tables de dimensions devraient toutes être « discrétisées » pour les rendre mieux exploitables par SSAS :

|  |  |
| --- | --- |
| Informations brutes | Informations discrétisées |
| Age du client « physique » ou « moral » | Tranches d’âges par 5 ans, entre 0 et 80 ans (Si plus : Inclus dans la dernière tranche) |
| Pourcentage de réduction fidélité du client | Tranches de pourcentages par 5 % entre 0 et 100 % |
| Date d’ouverture du magasin | Année d’ouverture |
| Surface du magasin | 3 classes : [0 .. 200] , [200 .. 2000] , [2000 .. + INF] m² |
| Population d’une ville | 5 classes : [0 .. 5 000] , [5 000 .. 10 000] , [10 000 .. 50 000] , [50 000 .. 200 000] , [200 000 .. +INF] habitants |

Par extension, seules les tables de faits devraient contenir des informations continues, comme les quantités ou les montants de vente et de marge.

## Impact de la hiérarchie des dimensions

Dans une dimension donnée, toutes les informations non-liées (cad. Contenant des informations qui sont fonctionnellement différentes) sont des axes d’analyse à part entière. Par extension, toutes les informations liées peuvent être obtenues à partir de leur parent(s)

* ***Exemple 1****: Ville ⇒ Un unique département ⇒ Une unique région, ces informations sont donc liées entre elles, et ne constituent pas deux axes fonctionnels d’analyse.*
* ***Exemple 2****: Age client & villes clients ne sont pas liées entre elles,, car il n’y a pas de unicité entre l’âge d’un client et sa ville. Ces informations constituent donc deux axes fonctionnels d’analyse.*

On distinguera donc deux types d’agrégats :

* **Agrégats « fondamental »** : Toutes les informations qui le composent ne sont pas liées entre elles.
* **Agrégats « d’optimisation »**: Les informations qui le compose sont liées entre elles, mais leur calcul de somme « à la volée » nécessiterait trop de ressources et pénaliserait donc significativement les performances des requête MDX.

# Annexe

## Lien MSDN ”Multidimensional Model Programming”

<https://msdn.microsoft.com/en-us/library/hh230848(v=sql.120).aspx>

## Lien MSDN “ADOMD.NET Client Programming”

<https://technet.microsoft.com/en-us/library/ms123477(v=sql.120).aspx>

## Script de volumétrie de l’entrepôt de données

/\*=====================================================================================

Fichier: Script\_Volumetrie\_DWH.sql

Résumé: Réalise le comptage du cube sur les dimensions {CLIENT, LIEU, DATE, PRODUIT}

Date: 02/08/2015

Updated:

SQL Server Version: 2014

======================================================================================\*/

-->> NOTE: THIS SCRIPT MUST BE RUN IN SQLCMD MODE INSIDE SQL SERVER MANAGEMENT STUDIO. <<--

-- Menu "QUERY" > "SQLCMD MODE"

:on error exit

-- Connexion à la base

USE [DataWarehouseODE];

SET nocount ON;

GO

select V.CLIENT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.CLIENT\_FK

print 'Nombre de tuples {Client} : ' + cast(@@ROWCOUNT as varchar);

select V.LIEU\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.LIEU\_FK

print 'Nombre de tuples {Lieu} : ' + cast(@@ROWCOUNT as varchar);

select V.DATE\_VENTE\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.DATE\_VENTE\_FK

print 'Nombre de tuples {Date} : ' + cast(@@ROWCOUNT as varchar);

select V.PRODUIT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.PRODUIT\_FK

print 'Nombre de tuples {Produit} : ' + cast(@@ROWCOUNT as varchar);

select V.CLIENT\_FK, V.LIEU\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.CLIENT\_FK, V.LIEU\_FK

print 'Nombre de tuples {Client – Lieu} : ' + cast(@@ROWCOUNT as varchar);

select V.CLIENT\_FK, V.DATE\_VENTE\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.CLIENT\_FK, V.DATE\_VENTE\_FK

print 'Nombre de tuples {Client – Date} : ' + cast(@@ROWCOUNT as varchar);

select V.CLIENT\_FK, V.PRODUIT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.CLIENT\_FK, V.PRODUIT\_FK

print 'Nombre de tuples {Client – Produit} : ' + cast(@@ROWCOUNT as varchar);

select V.LIEU\_FK, V.DATE\_VENTE\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.LIEU\_FK, V.DATE\_VENTE\_FK

print 'Nombre de tuples {Lieu – Date} : ' + cast(@@ROWCOUNT as varchar);

select V.LIEU\_FK, V.PRODUIT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.LIEU\_FK, V.PRODUIT\_FK

print 'Nombre de tuples {Lieu – Produit} : ' + cast(@@ROWCOUNT as varchar);

select V.DATE\_VENTE\_FK, V.PRODUIT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.DATE\_VENTE\_FK, V.PRODUIT\_FK

print 'Nombre de tuples {Date – Produit} : ' + cast(@@ROWCOUNT as varchar);

select V.CLIENT\_FK, V.DATE\_VENTE\_FK, V.LIEU\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.CLIENT\_FK, V.DATE\_VENTE\_FK, V.LIEU\_FK

print 'Nombre de tuples {Client – Date – Lieu} : ' + cast(@@ROWCOUNT as varchar);

select V.LIEU\_FK, V.PRODUIT\_FK, V.CLIENT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.LIEU\_FK, V.PRODUIT\_FK, V.CLIENT\_FK

print 'Nombre de tuples {Lieu – Produit – Client} : ' + cast(@@ROWCOUNT as varchar);

select V.DATE\_VENTE\_FK, V.PRODUIT\_FK, V.LIEU\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.DATE\_VENTE\_FK, V.PRODUIT\_FK, V.LIEU\_FK

print 'Nombre de tuples {Date – Produit – Lieu} : ' + cast(@@ROWCOUNT as varchar);

select V.DATE\_VENTE\_FK, V.PRODUIT\_FK, V.CLIENT\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.DATE\_VENTE\_FK, V.PRODUIT\_FK, V.CLIENT\_FK

print 'Nombre de tuples {Date – Produit – Client} : ' + cast(@@ROWCOUNT as varchar);

select V.LIEU\_FK, V.PRODUIT\_FK, V.CLIENT\_FK, V.DATE\_VENTE\_FK, COUNT(\*) from ODE\_DATAWAREHOUSE.FACT\_VENTES V group by V.LIEU\_FK, V.PRODUIT\_FK, V.CLIENT\_FK, V.DATE\_VENTE\_FK

print 'Nombre de tuples {Lieu – Produit – Client - Date} : ' + cast(@@ROWCOUNT as varchar);

GO

## Reports SSRS pour les métiers de la Direction

Evolution (du CA ; de la marge) ⊗ (par mois ; par année ; par département)

* + **Report R1** : Evolution CA et marge par mois sur la France entière
  + **Report R2** : Evolution CA et marge par année sur la France entière
  + **Report R3** : Evolution CA et marge par mois par département
  + **Report R4** : Evolution CA et marge par année par département

## Reports SSRS pour les métiers du Marketing

Evolution **(**du volume des ventes) ⊗ (par mois ; par département ; par type de produit)

* + **Report R5** : Evolution du vol des ventes par mois sur la France entière par rayon de produit
  + **Report R6** : Evolution du vol des ventes par mois sur la France entière par famille de produit
  + **Report R7** : Evolution du vol des ventes par mois par département et par rayon de produit
  + **Report R8** : Evolution du vol des ventes par mois par département et par famille de produit

Evolution **(**du taux de percée) ⊗ (par mois ; par département ; par magasin)

* + **Report R9** : Evolution du taux de percée par mois et par magasin
  + **Report R10** : Evolution du taux de percée par mois et par département

|  |
| --- |
| Taux de percée  Sur une période donnée, on le définit comme le % de ratio de la population d’une ville client du magasin sur cette période, par rapport à la population totale de la ville.  *Exemple : Au mois de mai 2015, le magasin de TALENCE a eu 1 234 clients de la ville voisine de PESSAC, qui compte 12 340 habitants.*  *Son taux de percée est donc de 10 % en mai 2015.* |