UV : BD51

Rapport projet BD51 :
Business Intelligence & Data Warehouse

Groupe 17 :

BLLADI Ismail

DRISSI SLIMANI Youness

EL YASSAMI Hafsa

Table des matières

Table des matières 1
Introduction 7
Partie 1 : implémentation des fonctions ETL 8
I. Qualité des données 8
1. Description de la Base de données EMODE 8
2. Vérification des données 8
II. Développement des package SSIS 11
1. Package 1 11
2. Package 2 17
3. Package 3 27
4. Package 4 33
5. Test ETL 34
6. Automatisation de l'exécution des packages 39
Partie 2 : Optimisation du Data Warehouse 44
III. Partitionnement de la table de faits 44
1. Mise en œuvre du partitionnement 45
IV. Projet Analysis Services 47
1. Source de données 47
2. Calcul nommé 48
3. Dimensions 50
4. Cube et mesures 54
5. Exploration du cube 55
Partie 3 : La mise en place du reporting 56
V. Reporting services 56
VI. Univers Busniess Objects 59
1. Création de l'univers 59
2. Navigation agrégée 61
VII. Web Intelligence 62
1. Premier Document : 62
2. Second Document : 64
3. Ajout des contrôles de saisie 66

VIII. Excel 67
1. Rapports 67
2. Power pivot 70
IX. QlikView 73
1. Chargement des données 73
2. Tableau de bord 75
Conclusion 76

Table de Figures

Figure 1 : Script vérification clé primaire 1 9
Figure 2 : script SQL vérification clé primaire 2 9
Figure 3 : script vérification clé étranger 10
Figure 4 : script création table TREJ 11
Figure 5 : Script suppression des clés étrangères 12
Figure 6 : Script TRUNCATE TABLE 13
Figure 7 : Script ajout clés étrengères 13
Figure 8 : Vue globale Package 1 14
Figure 9 : Vérification des PK dans AL 15
Figure 10 : Data Flow transfert de données AL 15
Figure 11 : Extrait du DATA FLOW "SHOP_FACTS 16
Figure 12 : Les tables de EMODE_INC 17
Figure 13 : Script création ARTICLE_LOOKUP_INC 17
Figure 14 : Liste des triggers 18
Figure 15 : Script de test d'insertion CALENDAR_YEAR_LOOKUP 18
Figure 16 : Insertion CALENDAR_YEAR_LOOKUP_INC 18
Figure 17 : Exemple script Trigger d'insertion 21
Figure 18 : Script d'accorder des droits 22
Figure 19 : Script de création AUDIT_TRACE 23
Figure 20 : Table AUDIT_TRACE 23
Figure 21 : Script de création AUDIT_DETAILS 23
Figure 22 : Table AUDIT_DETAILS 24
Figure 23 : Script de création AUDIT_ERRORS 24
Figure 24 : Vue Globale package 2 - partie 1 26
Figure 25 : Vue Globale package 2 - partie 2 26
Figure 26 : Script de création AUDIT_AUTO_ERRORS 29
Figure 27 : Table AUDIT_AUTO_TRACE 29
Figure 28 : Script de création AUDIT_AUTO_TRACE 28
Figure 29 : Table AUDIT_AUTO_ERRORS 29
Figure 30 : Variables Row count 31

Figure 31 : Vue Globale package 3 32
Figure 32 : Vue global package 4 33
Figure 33 : Data Flow du package 4 34
Figure 34 : Exécution du package 1 35
Figure 35 : Exécution du transfert de OUTLET_LOOKUP - Package 1 36
Figure 36 : Ajout des lignes sur EMODE_INC 36
Figure 37 : Exécution du package 2 36
Figure 38 : Exécution du Data Flow CYL 37
Figure 39 : Ajout des lignes sur SQL SERVER 37
Figure 40 : Exécution du package 3 38
Figure 41 : Exécution Data Flow SF 38
Figure 42 : Exécution du package 4 39
Figure 43 : ETL Exécution immédiate 40
Figure 44 : Exécution du package 3 CMD 40
Figure 45 : Tâche planifiée Windows 41
Figure 46 : Création du Job SQL Server Agent 42
Figure 47 : Création du Job Audit Package 2 42
Figure 48 : Création du planificateur Job Schedule 43
Figure 49 : Script de création du storage 45
Figure 50 : Script d'association au fichier physique 45
Figure 51 : Script de fonction de partition 46
Figure 52 : Schéma de partition 46
Figure 53 : Script de création de table TEMP 46
Figure 54 : Script d'ajout de contrainte 47
Figure 55 : Figure - Source de données cube OLAP 48
Figure 56 : Figure Propriété MONTH YEAR 49
Figure 57 : Propriété WEEK 49
Figure 58 : Attributs et hiérarchie dimension "Time" 50
Figure 59 : Navigation dimension "Time" 50
Figure 60 : Attribut et hiérarchie dimension "Geography" 51
Figure 61 : Navigation dimension "Geography" 51
Figure 62 : Attributs et hiérarchie dimension "Article" 52

Figure 63 : Navigation dimension "Article" 52
Figure 64 : Attributs et hiérarchie dimension "ArticleColor" 53
Figure 65 : Navigation dimension "ArticleColor" 53
Figure 66 : Mesures du cube 54
Figure 67 : Mesures précalculées 54
Figure 68 : CA par année et par magasin 55
Figure 69 : SSRS Rapport 1 Table croisé 1 56
Figure 70 : SSRS Rapport 1 Table croisé 2 57
Figure 71 : SSRS Rapport 2 Table et graphe 58
Figure 72 : SSRS Rapport 3 Table et graphe 59
Figure 73 : Univers modèle en étoile 60
Figure 74 : Structure de l'univers 61
Figure 75 : Exemple d'agrégation 62
Figure 76 : WI Doc 1 Rapport 1 62
Figure 77 : WI Doc 1 Rapport 2 63
Figure 78 : WI Doc 1 Page de garde 63
Figure 79 : WI Doc 2 Rapport 1 64
Figure 80 : WI Doc 2 Rapport 2 65
Figure 81 : WI Doc 2 Rapport 3 65
Figure 82 : WI Doc 1 Rapport 1 avec filtre 66
Figure 83 : WI Doc 1 Rapport 2 avec filtre 66
Figure 84 : Modèle de données sous EXCEL 67
Figure 85 : EXCEL Table simple et camembert 68
Figure 86 : EXCEL Table croisé et graphe 68
Figure 87 : EXCEL Table d'indicateurs 69
Figure 88 : EXCEL Règles de la table d'indicateurs 69
Figure 89 : EXCEL Tableau de bord 70
Figure 90 : POWER PIVOT ventes par pays 1 71
Figure 91 : POWER PIVOT ventes par pays 2 71
Figure 92 : POWER PIVOT Rapport 2 72
Figure 93 : POWER PIVOT Rapport 3 73
Figure 94 : Qlikview script ACL 74

Figure 95 : Qlikview modèle 74
Figure 96 : Qlikview tableau de bord 75
Figure 97 : Qlikview tableau de bord avec filtre par mois 75

Introduction

Situé dans un monde où l'information est devenue une grande richesse si elle est bien exploitée, l'informatique décisionnelle parvient à l'exploitation même d'un volume intéressent des données en tout efficacités. Elle permet de précipiter et d'améliorer la prise de décision, d'optimiser les processus, d'accroitre l'efficience d'exploitation.

Dans le cadre de l'Unité de Valeur BD51 on est amené à la réalisation d'un système de Business Intelligence qui commence par la collecte, l'analyse puis la restitution et finalement la modélisation des données. Ce système est conçu afin de permettre une gestion efficiente d'un magasin en toute simplicité pour les personnes non familiarisées avec l'informatique.

Notre travail est reparti en 3 grandes parties, dont la partie initiale est l'implémentation des fonctions ETL, puis l'optimisation du Data Warehouse ainsi que l'élaboration des différents rapports. Tout le processus est bien détaillé jusqu'à la mise en place de cette solution.

Pendant ce travaille nous avons utilisé la base de données EMODE qui se compose des informations de ventes de différents magasins, ces données se rapprochent le plus possible des données réelles.

Ainsi ce document est conçu afin d'apporter une explication explicite sur les approches engagées pour atteindre un outil d'aide à lé décision, pertinent s'approchant au plus près des besoins réels.

Partie 1 : implémentation des fonctions ETL

I. QUALITE DES DONNEES

1. Description de la Base de données EMODE

Le projet réside sur une base de données en modèle étoile, qui se compose de 5 tables dont la table SHOP_FACT est la table de fait et les autres tables correspond aux dimensions (ARTICLE_COLOR_LOOKUP, ARTICLE_LOOKUP, OUTLET_LOOKUP…)

Les abréviations utilisées:

ACL: ARTICLE_COLOR_LOOKUP
AL: ARTICLE_LOOKUP
OL: OUTLET_LOOKUP
CYL: CALENDAR_YEAR_LOOKUP
SH: SHOP_FACTS

2. Vérification des données

L 'absence des contrainte d'intégrité dans la base EMODE implique que vérification des données est indispensable afin de réaliser un transfert de données d'Oracle vers SQL Server et de garantir la qualité des données initiales. Afin d'assurer la cohérence de données, des requêtes SQL sont mises en place pour examiner l'unicité des clés primaire ainsi que l'existence des clés étrangère dans les tables mères, sans oublier les valeurs de ces derniers qui doivent différés de NULL.

a. Unicité de la clé primaire

La clé primaire est un identifiant unique d'une ligne d'une table, son existence et son unicité sont les piliers pour avoir une base de données cohérente. La vérification est faite à travers des requêtes SQL sur chaque table indépendamment qui sera présenté dans les figures qui suit.

- CALENDAR_YEAR_LOOKUP:

```
-- -------------------- --
-- CALENDAR_YEAR_LOOKUP --
-- -------------------- --
SELECT
   CYL.WEEK_KEY "Week key"
   ,COUNT(CYL.WEEK_KEY) "Nombre de clés primaires"
FROM
   CALENDAR_YEAR_LOOKUP CYL
GROUP BY
   CYL.WEEK_KEY
HAVING
   COUNT(CYL.WEEK_KEY) > 1
ORDER BY
   1 ASC
;
```

Figure 1 : Script vérification clé primaire 1

Pout les tables OUTLET_LOOKUP, ARTICLE_LOOKUP le traitement était pareil à la table CALENDAR_YEAR_LOOKUP, on a utilisé une requête SQL qui nous permet de sélectionner les clés primaires doublant.

- ARTICLE_COLOR_LOOKUP:

```
-- -------------------- --
-- ARTICLE_COLOR_LOOKUP --
-- -------------------- --
SELECT
   ACL.ARTICLE_CODE AS "Article code"
   ,ACL.COLOR_CODE AS "Color code"
   ,CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE) AS "Article_Color_Code"
   ,COUNT(CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE)) AS "Nombre de clés
primaires"
FROM
   ARTICLE_COLOR_LOOKUP ACL
GROUP BY
   ACL.ARTICLE_CODE
   ,ACL.COLOR_CODE
   ,CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE)
HAVING
   COUNT(CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE))  > 1
ORDER BY
   1 ASC
;
```

Figure 2 : script SQL vérification clé primaire 2

Cette table, on a une clé primaire composé, ce qui explique un traitement différent mais qui réside sur le même principe que les autres.

b. Existence des clés étrangères :

Cependant la table SHOP_FACTS comme étant la table de faits, elle dispose de toutes les clés étrangères et la vérification donc est faite au niveau de l'existence de ces derniers dans leur table d'origine

- SHOP_FACTS :

```sql
-- ---------------------------- --
-- ARTICLE_LOOKUP -> SHOP_FACTS --
-- ---------------------------- --
SELECT
   ARTICLE_CODE
FROM
   SHOP_FACTS
WHERE
   ARTICLE_CODE
NOT IN (SELECT ARTICLE_CODE FROM ARTICLE_LOOKUP);

-- ---------------------------------- --
-- ARTICLE_COLOR_LOOKUP -> SHOP_FACTS --
-- ---------------------------------- --
SELECT
   DISTINCT ARTICLE_CODE||COLOR_CODE
FROM
   SHOP_FACTS
WHERE
   ARTICLE_CODE||COLOR_CODE
NOT IN (SELECT ARTICLE_CODE||COLOR_CODE FROM ARTICLE_COLOR_LOOKUP);

-- ---------------------------------- --
-- CALENDAR_YEAR_LOOKUP -> SHOP_FACTS --
-- ---------------------------------- --
SELECT
   WEEK_KEY
FROM
   SHOP_FACTS
WHERE
   WEEK_KEY
NOT IN (SELECT WEEK_KEY FROM CALENDAR_YEAR_LOOKUP);

-- --------------------------- --
-- OUTLET_LOOKUP -> SHOP_FACTS --
-- ---------------------------- --
SELECT
   SHOP_CODE
FROM
   SHOP_FACTS
WHERE
   SHOP_CODE
NOT IN (SELECT SHOP_CODE FROM OUTLET_LOOKUP);
```

Figure 3 : script vérification clé étranger

II. DEVELOPPEMENT DES PACKAGE SSIS

Afin d'assurer un transfert de données sain, on a créé des packages qui vont gérer ce déplacement en respectant toutes les contraintes afin d'avoir une base de données cohérente.

Au niveau des outils, on a utilisé *SQL Server Integration Services* (*SSIS*) dans *SQL Server Data Tools* (*SSDT*).

1. Package 1

L'objectif du package 1 est le transfert intégral des données existantes dans la base ORACLE vers la base SQL SERVER dont le processus inclut la suppression des données dans les tables de la base de données destination. Ainsi on considère que les tables du modèle en étoile existent dans la base SQLSERVER.

Initialement, nous avons exploité le gestionnaire de connexion pour créer deux connexions, la première pour assurer la connexion au schéma coté Oracle et la deuxième pour assurer la connexion coté SQL Server.

a. Tables de rejets

Au cours du transfert des données de la base source vers la base destination, les lignes qui ne respectent pas les contraintes seront enregistrer dans les tables de rejets. Chaque table destination a son équivalent en rejet dont la forme des noms est <NOM_TABLE>_TREJ. Ainsi on a créé 5 tables avec le même principe selon ce script :

```
-- ---------------------------- --
---------- CREAT TABLE TREJ ------
CREATE TABLE ARTICLE_COLOR_LOOKUP_TREJ
AS (SELECT *
    FROM
        ARTICLE_COLOR_LOOKUP
    WHERE
        0=1
)
;
```

Figure 4 : script création table TREJ

Ces tables sont conçues afin de garder des traces des données incohérentes. Ainsi il faut noter que ces tables seront utilisées tout au long du projet pour les différents packages.

b. L'enchainement du processus

L'arrangement du package 1 est le suivant :

- Désactivation des contraintes de clés étrangères dans la table de destination SHOP_FACTS.
- Suppression des données des 5 tables de destination.
- Réactivation des contraintes de clés étrangères dans la table de destination SHOP_FACTS.
- Transfert des données des tables de dimension.
- Transfert des données de la table SHOP_FACTS.

Désactivation des FK :

Cette étape est primordiale pour la suite du processus, sans la désactivation des clés étrangères la suppression des données qui est l'étape suivante ne pourra être réaliser.

La désactivation des clés étrangers peut se faire à l'aide de requêtes SQL suivantes :

```
-- ---------------------------- --
---------- Drop FK --------------

--Drop ARTICLE_COLOR_LOOKUP FK
ALTER TABLE SHOP_FACTS
DROP CONSTRAINT FK_SHOP_FACTS_ARTICLE_COLOR_LOOKUP;
--Drop ARTICLE_LOOKUP FK
ALTER TABLE SHOP_FACTS
DROP CONSTRAINT FK_SHOP_FACTS_ARTICLE_LOOKUP;
--Drop OUTLET_LOOKUP FK
ALTER TABLE SHOP_FACTS
DROP CONSTRAINT FK_SHOP_FACTS_OUTLET_LOOKUP;
--Drop CALENDAR_YEAR_LOOKUP FK
ALTER TABLE SHOP_FACTS
DROP CONSTRAINT FK_SHOP_FACTS_CALENDAR_YEAR_LOOKUP;
```

Figure 5 : Script suppression des clés étrangères

TURNCATE des tables :

La suppression des données est faite en utilisant l'instruction TURNCATE, le choix de l'instruction a été fait du a ses avantages face à l'instruction DELETE qui se résument en plusieurs niveaux.

TURNCATE utilise moins de verrous que DELETE, elle permet la réinitialisation de la valeur d'une colonne avec la propriété auto-incrémentation, ainsi en termes de ressources elle est plus optimale par conséquence est plus rapide sans oublier qu'elle ne déclenche pas de TRIGGER.

A travers les requêtes SQL suivantes, les tables de destination (la table de *faits*, les tables de dimension et leurs équivalents de rejet) seront toutes purger.

```
-- ---------------------------- --
-- Suppression des données
-- ---------------------------- --
TRUNCATE TABLE SHOP_FACTS;
TRUNCATE TABLE OUTLET_LOOKUP;
TRUNCATE TABLE ARTICLE_COLOR_LOOKUP;
TRUNCATE TABLE ARTICLE_LOOKUP;
TRUNCATE TABLE CALENDAR_YEAR_LOOKUP;
```

Figure 6 : Script TRUNCATE TABLE

Réactivation des FK :

Afin d'avoir un transfert de données sain tout en respectant les règles d'intégrités. Il est nécessaire de réintégrer les clés étrangères après l'exécution des requêtes TURNCATE, ainsi on commence le transfert de données.

```
-- ---------------------------- --
--Add FK in SHOP_FACTS
-- ---------------------------- --
--Link with ARTICLE_COLOR_LOOKUP
ALTER TABLE SHOP_FACTS WITH CHECK
ADD CONSTRAINT FK_SHOP_FACTS_ARTICLE_COLOR_LOOKUP FOREIGN KEY
(ARTICLE_CODE, COLOR_CODE)
REFERENCES ARTICLE_COLOR_LOOKUP(ARTICLE_CODE, COLOR_CODE);

--Link with ARTICLE_LOOKUP
ALTER TABLE SHOP_FACTS WITH CHECK
ADD CONSTRAINT FK_SHOP_FACTS_ARTICLE_LOOKUP FOREIGN KEY (ARTICLE_CODE)
REFERENCES ARTICLE_LOOKUP(ARTICLE_CODE);

--Link with OUTLET_LOOKUP
ALTER TABLE SHOP_FACTS WITH CHECK
ADD CONSTRAINT FK_SHOP_FACTS_OUTLET_LOOKUP FOREIGN KEY (SHOP_CODE)
REFERENCES OUTLET_LOOKUP(SHOP_CODE);

--Link with CALENDAR_YEAR_LOOKUP
ALTER TABLE SHOP_FACTS WITH CHECK
ADD CONSTRAINT FK_SHOP_FACTS_CALENDAR_YEAR_LOOKUP FOREIGN KEY (WEEK_KEY)
REFERENCES CALENDAR_YEAR_LOOKUP(WEEK_KEY);
```

Figure 7 : Script ajout clés étrengères

Pour le transfert des données, chaque table sera présentée par un DATA FLOW, ce qui permet la lecture du package ainsi que sa compréhensibilité. Cependant l'ordre du

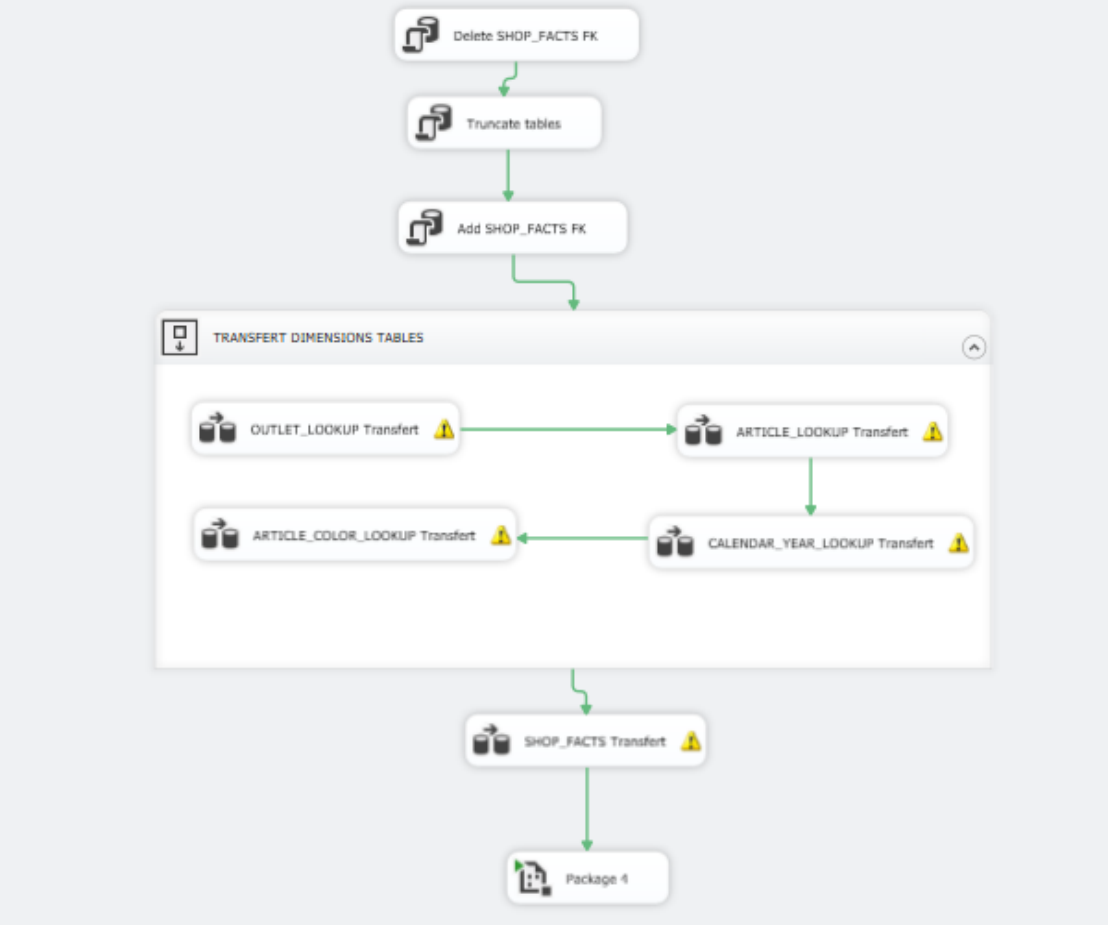


Figure 8 : Vue globale Package 1

transfère est essentiel, on commencer par les tables de dimension ensuite la table des faits.

c. Fonctionnement Data Flow

Tables des dimensions :

Les démarches pour le transfert de données sont identiques pour les quatre tables. On commence par l'extraction des données de la base EMODE, ensuite la vérification de l'intégrité de ces derniers avant de les copier dans la base de données de destination. Afin d'assurer la qualité des données, on a utilisé la transformation LOOKUP pour pouvoir acheminer les données vers la base destination, vers la table qui correspond s'ils respectent les contraintes sinon ils seront redirigés vers les tables de rejets.

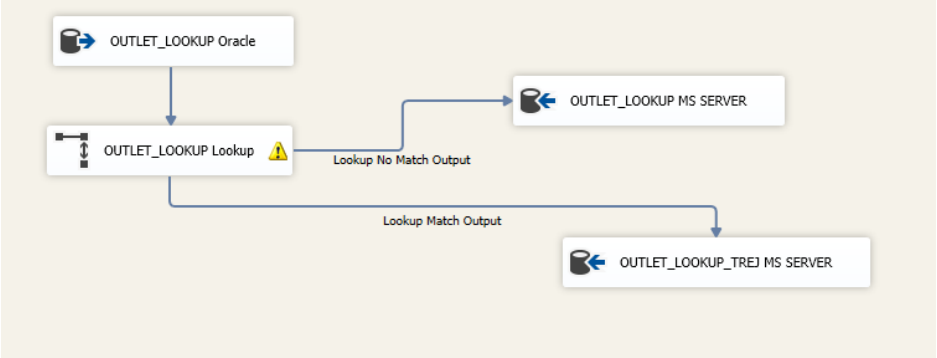


Figure 10 : Data Flow transfert de données AL

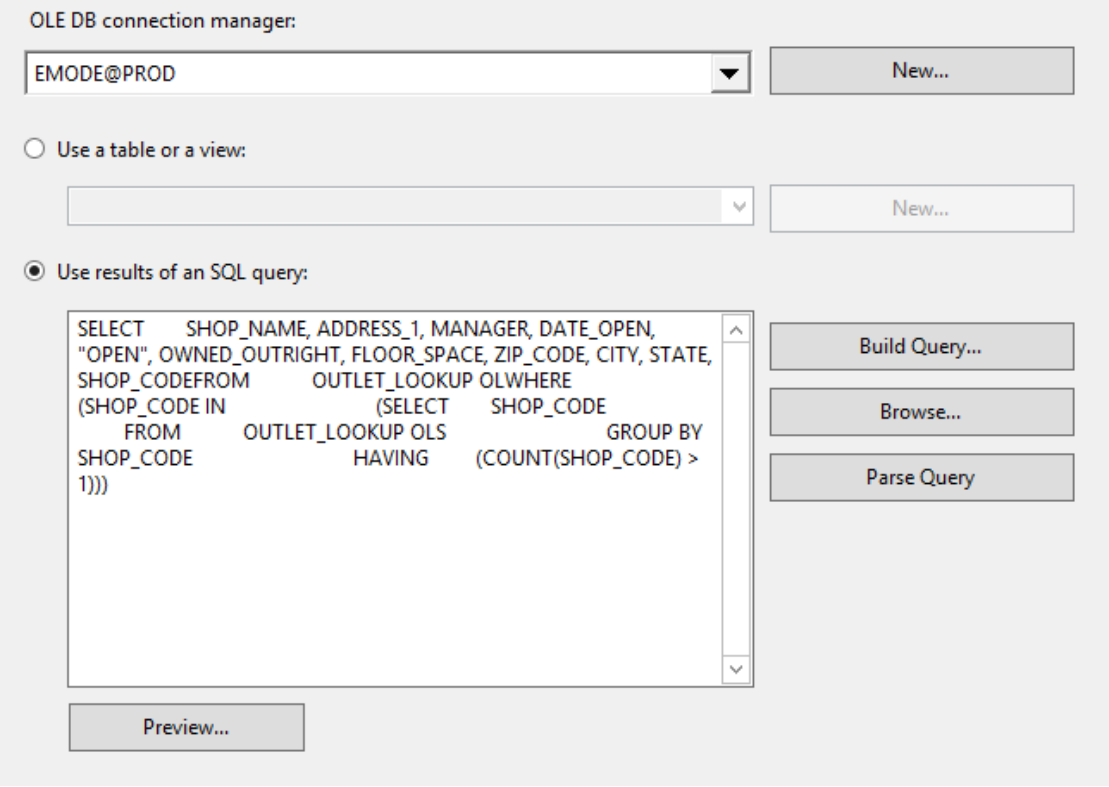


Figure 9 : Vérification des PK dans AL

Table de faits :

On commence tout abord par la vérification de l'existence des clés étrangers dans les tables de dimension, dans la base de destination. Pour cela la transformation LOOKUP ne sera pas fonctionnel puisqu'elle permet de faire une équijointure entre des données issues d'une même source, ce qui n'est pas le cas pour notre table de faits. Afin de comparer les données issues de différentes sources, on fait appel à la transformation SSIS « MERGE JOIN ». Pour le transfert des données de la table SHOP_FACTS, on a réalisé une jointure externe à gauche en faveur des tables EMODE sous ORACLE, ceci sera appliquer pour la vérification des quatre tables de dimension.

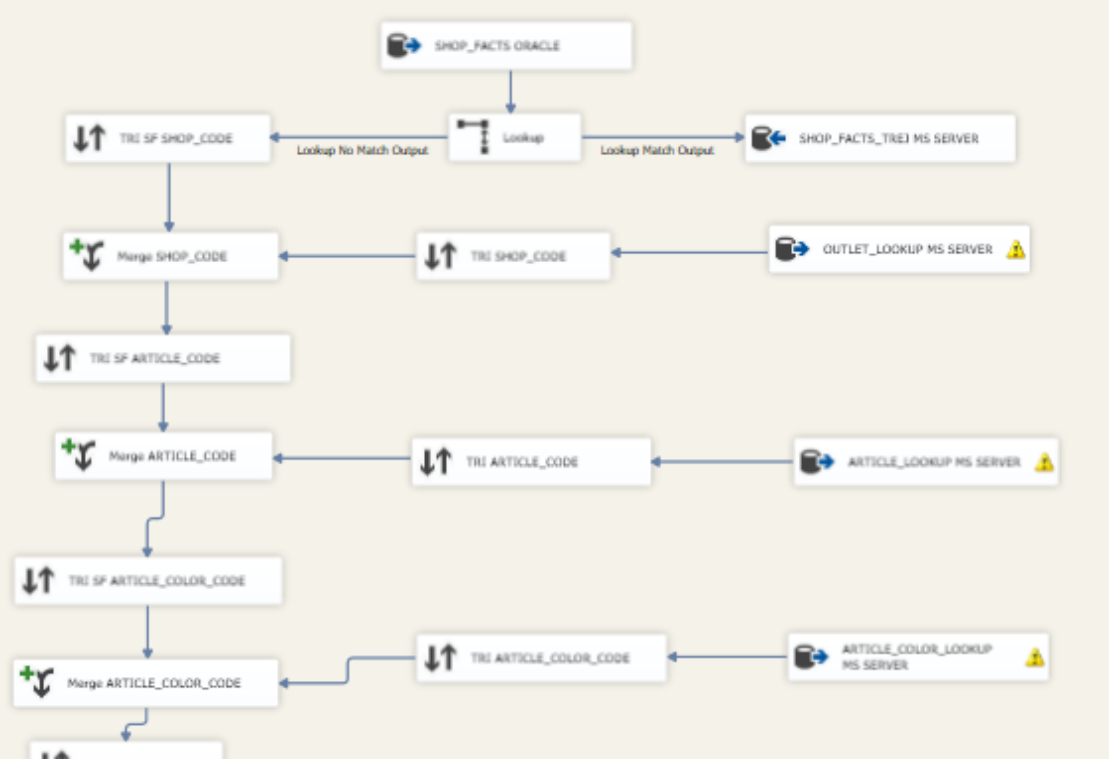


Figure 11 : Extrait du DATA FLOW "SHOP_FACTS

À chaque jointure externe sur les clés étrangères de la table de faits sous Oracle et les clés primaires des dimensions sous SQL SERVER, on ajoute une colonne avec la clé primaire de la dimension.

2. Package 2

Ce deuxième package aura 2 fonctionnalités, premièrement assuré un transfert incrémental des données, puis deuxièmement avoir une traçabilité de ce transfert à travers les tables d'audit.

a. Transfert incrémental

Schéma EMODE_INC

Pour ce transfère incrémental, on crée un nouveau schéma nommé EMODE_INC, dont la structure des tables est pareille que celle de EMODE avec des noms de la forme <NOM_TABLE>_INC. Ce nouveau schéma va contenir les dernières modifications apportées à EMODE pour éviter de transférer toutes les données à chaque exécution.

Figure 12 : Les tables de EMODE_INC

Les tables dans EMODE_INC auront la même structure que celle de EMODE mais on leur ajoute un champ supplémentaire « OPERATION_TYPE » qui va permettre d'identifier l'opération de modification qui a été faite, ce champ peut prendre trois valeurs :

- « INSERT » pour une insertion
- « UPDATE » pour une mise à jour
- « DELETE » pour une suppression

Vous trouvez ci-dessous le script de création de la table ARTICLE_LOOKUP_INC

```
--Create table ARTICLE_LOOKUP_INC
CREATE TABLE ARTICLE_LOOKUP_INC
(
ARTICLE_CODE NUMBER (6),
ARTICLE_LABEL VARCHAR2 (45),
CATEGORY VARCHAR2 (25),
SALE_PRICE NUMBER(8,2),
FAMILY_NAME VARCHAR2 (20),
FAMILY_CODE VARCHAR2 (3),
OPERATION_TYPE VARCHAR2 (6),
CONSTRAINT CHECK_OPERATION_TYPE_AL CHECK (OPERATION_TYPE IN('INSERT','UP-
DATE','DELETE'))
);
```

Figure 13 : Script création ARTICLE_LOOKUP_INC

Alimentation de EMODE_INC

Afin d'alimenter EMODE_INC, nous allons utiliser des « Trigger » sur les tables de EMODE pour copier les données dans les tables correspondantes dans EMODE_INC. Le fonctionnement de ces derniers sera déclencher après une insertion, une mise à jour ou une suppression.

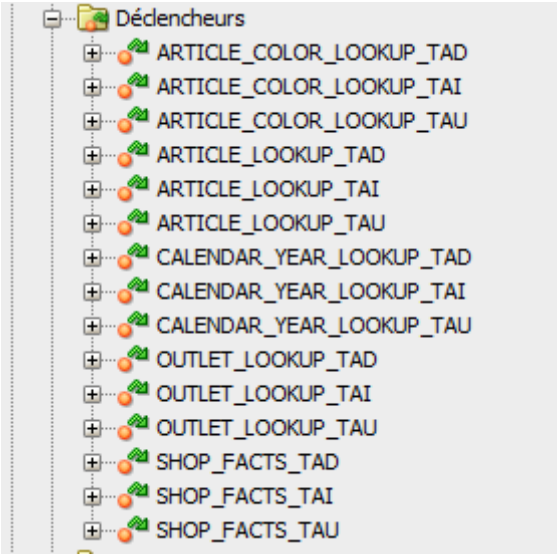


Figure 14 : Liste des triggers

Chacune de ces actions sur une table dans EMODE va générer une ligne dans la table correspondante dans EMODE_INC avec le flag identifiant l'opération qui a eu lieu. Par exemple lors d'insertion dans CALENDAR_YEAR_LOOKUP on a une ligne qui s'ajoute dans la table CALENDAR_YEAR_LOOKUP_INC :

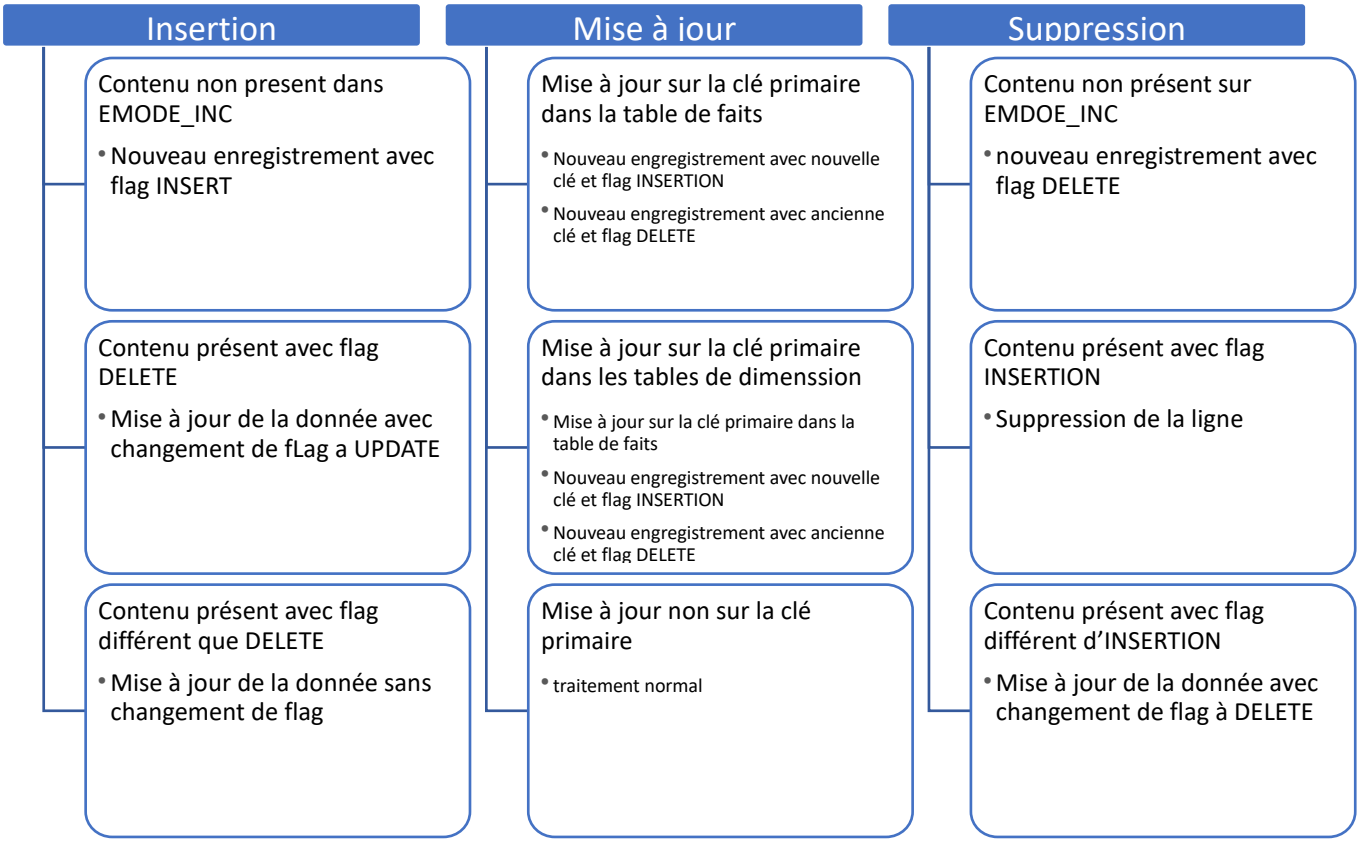
```
1  -- -------------------- --
2  -- CALENDAR_YEAR_LOOKUP --
3  -- -------------------- --
4
5  INSERT INTO CALENDAR_YEAR_LOOKUP
6      (WEEK_KEY,WEEK_IN_YEAR,YEAR,FISCAL_PERIOD,YEAR_WEEK,QUARTER,MONTH_NAME,MONTH,HOLIDAY_FLAG)
7  VALUES
8      (263,1,2018,'FY18','2018/1',1,'January',1,'y');
9
10 INSERT INTO CALENDAR_YEAR_LOOKUP
11     (WEEK_KEY,WEEK_IN_YEAR,YEAR,FISCAL_PERIOD,YEAR_WEEK,QUARTER,MONTH_NAME,MONTH,HOLIDAY_FLAG)
12 VALUES
13     (264,2,2018,'FY18','2018/2',1,'January',1,'n');
14
```

Figure 15 : Script de test d'insertion CALENDAR_YEAR_LOOKUP

	WEEK_KEY	WEEK_IN_YEAR	YEAR	FISCAL_PERIOD	YEAR_WEEK	QUARTER	MONTH_NAME	MONTH	HOLIDAY_FLAG	OPERATION_TYPE
1	263	1	2018	FY18	2018/1	1	January	1	y	INSERT
2	264	2	2018	FY18	2018/2	1	January	1	n	INSERT

Figure 16 : Insertion CALENDAR_YEAR_LOOKUP_INC

Le développement des « triggers » nous permet de traiter tous les scénarii possibles. Les schémas suivants récapitulent tous les cas pris en compte par les triggers :



Certains cas peuvent nécessiter plus d'explication, comme dans le cas d'une suppression, si un enregistrement est déjà présent pour cette clé primaire avec le flag « INS », il n'est pas nécessaire de mettre à jour cet enregistrement. Si le flag « INS » est présent, cela veut dire que le package n'a pas encore été exécuté et que l'enregistrement n'est pas présent dans la table de destination, on peut donc simplement le supprimer du schéma EMODE_INC.

Le schéma EMODE ne comporte aucune contrainte d'intégrité donc il peut arriver que la clé primaire soit modifiée. Si on insère simplement un enregistrement dans EMODE_INC avec la nouvelle clé primaire et le flag « UPD », lors de l'exécution du package une erreur sera générée car nous allons exécuter une requête de mise à jour sur une clé qui n'existe pas encore. Pour pallier à ce problème, il faut donc ajouter un enregistrement avec l'ancienne clé primaire et le flag « DEL » et un deuxième enregistrement avec la nouvelle clé primaire et le flag « INS ».

```
-- After insert
CREATE OR REPLACE TRIGGER "EMODE"."SHOP_FACTS_TAI"
AFTER INSERT ON EMODE.SHOP_FACTS
FOR EACH ROW
DECLARE
	rowsdetected NUMBER(10);
	operationtype VARCHAR2(6);
BEGIN
	-- If PK is null then initialize idsf to 0
	IF(:new.ID IS NULL) THEN
		idsf := 0;
	ELSE
		idsf := :new.ID;
	END IF;
	-- Insert the number of rows existed in SHOP_FACTS_INC that have the PK
	SELECT
		COUNT(*) INTO rowsdetected
	FROM
		EMODE_INC.SHOP_FACTS_INC SFI
	WHERE
		SFI.ID = idsf;
	-- If no row then insert
	IF(rowsdetected=0) THEN
		INSERT INTO EMODE_INC.SHOP_FACTS_INC
		(
			ID
			,ARTICLE_CODE
			,COLOR_CODE
			,WEEK_KEY
			,SHOP_CODE
			,MARGIN
			,AMOUNT_SOLD
			,QUANTITY_SOLD
			,OPERATION_TYPE
		)
		VALUES
		(
			:new.ID
			,:new.ARTICLE_CODE
			,:new.COLOR_CODE
			,:new.WEEK_KEY
			,:new.SHOP_CODE
			,:new.MARGIN
```

```
                        ,:new.MARGIN
                        ,:new.AMOUNT_SOLD
                        ,:new.QUANTITY_SOLD
                        ,'INSERT'
                );
        -- Else Get operation type from SHOP_FACTS_INC
        ELSE
                SELECT
                        OPERATION_TYPE INTO operationtype
                FROM
                        EMODE_INC.SHOP_FACTS_INC SFI
                WHERE
                        SFI.ID = idsf;
         --IF opration type is 'DELETE' THEN change it to 'UPDATE' and update other col-
umns else update rows
                IF(operationtype='DELETE') THEN
                        UPDATE
                                EMODE_INC.SHOP_FACTS_INC
                        SET
                                ID = :new.ID
                                ,ARTICLE_CODE = :new.ARTICLE_CODE
                                ,COLOR_CODE = :new.COLOR_CODE
                                ,WEEK_KEY = :new.WEEK_KEY
                                ,SHOP_CODE = :new.SHOP_CODE
                                ,MARGIN = :new.MARGIN
                                ,AMOUNT_SOLD = :new.AMOUNT_SOLD
                                ,QUANTITY_SOLD = :new.QUANTITY_SOLD
                                ,OPERATION_TYPE = 'UPDATE'
                        WHERE
                                ID = idsf;
                ELSE
                        UPDATE
                                EMODE_INC.SHOP_FACTS_INC
                ELSE
                        UPDATE
                                EMODE_INC.SHOP_FACTS_INC
                        SET
                                ID = :new.ID
                                ,ARTICLE_CODE = :new.ARTICLE_CODE
                                ,COLOR_CODE = :new.COLOR_CODE
                                ,WEEK_KEY = :new.WEEK_KEY
                                ,SHOP_CODE = :new.SHOP_CODE
                                ,MARGIN = :new.MARGIN
                                ,AMOUNT_SOLD = :new.AMOUNT_SOLD
                                ,QUANTITY_SOLD = :new.QUANTITY_SOLD
                                ,OPERATION_TYPE = operationtype
                        WHERE
                                ID = idsf;
                END IF;
        END IF;

END SHOP_FACTS_TAI;
/

ALTER TRIGGER "EMODE"."SHOP_FACTS_TAI" ENABLE;
```

Figure 17 : Exemple script Trigger d'insertion

Dans le cadre du projet, on considère par exemple que si une modification intervient sur une clé primaire, il n'y a pas d'enregistrement correspondant à la nouvelle clé primaire dans EMODE_INC. Il faudrait alors ajouter un contrôle supplémentaire dans le « trigger ».

Au début des « triggers », nous testons si la clé primaire est « null ». Si c'est le cas, nous initialisons la valeur de la clé à « 0 ». Cette manière de procéder permet de gérer plus facilement les clés primaires « null ». Le fonctionnement des packages 2 et 3 ne change pas pour autant. Pour tester l'intégrité de la clé primaire, on vérifie que la clé est différente de « 0 ».

Droits sur EMODE_INC

Il a primordiale d'accorder certains droits sur les tables dans EMODE_INC, pour pouvoir l'alimenter à partir du schéma EMODE.

```
--Grant To emode

GRANT INSERT, UPDATE, DELETE ,SELECT
ON ARTICLE_COLOR_LOOKUP_INC
TO emode;

GRANT INSERT, UPDATE, DELETE ,SELECT
ON ARTICLE_LOOKUP_INC
TO emode;

GRANT INSERT, UPDATE, DELETE, SELECT
ON SHOP_FACTS_INC
TO emode;

GRANT INSERT, UPDATE, DELETE ,SELECT
ON OUTLET_LOOKUP_INC
TO emode;

GRANT INSERT, UPDATE, DELETE ,SELECT
ON CALENDAR_YEAR_LOOKUP_INC

TO emode;
```

Figure 18 : Script d'accorder des droits

b. Tables d'audits

Pendant l'exécution automatique d'un transfert, il est nécessaire de savoir quelques informations sur le déroulement de ce transfert ainsi des statistiques sur les données transférées, aussi pour archiver les erreurs survenues.

Chacune des trois tables mise en place pour l'audit de ce package à une utilité particulière.

- La table *AUDIT_TRACE* contiendra des informations d'ordre général comme la date du transfert, l'heure de début/de fin, un code erreur et un numéro unique pour identifier le transfert. Le code erreur est à 1 si au moins une erreur est survenue durant le transfert sinon il est à 0, ce qui permet d'avoir un aperçu global du déroulement du transfert.

```
CREATE TABLE AUDIT_TRACE
(
NUM_TRANSFER NUMERIC(15) IDENTITY(1,1) NOT NULL
,"DATE" DATE
,START TIME
,"END" TIME
,ERROR_CODE NUMERIC(1)
)
.
```

Figure 19 : Script de création AUDIT_TRACE

Results | Messages

	NUM_TRANSFER	DATE	START	END	ERROR_CODE
1	1	2019-01-03	00:19:19.8466667	00:19:31.1300000	0

Figure 20 : Table AUDIT_TRACE

- La table *AUDIT_DETAILS* comporte des informations statistiques sur le transfert, le nombre de lignes ajoutées, modifiées, supprimées et rejetées par table. NUM_TRANSFER fait référence au numéro du transfert correspondant dans *AUDIT_TRACE*.

```
CREATE TABLE AUDIT_DETAILS
(
NUM_DETAIL NUMERIC(10) NOT NULL IDENTITY(1,1)
,TABLE_NAME VARCHAR(25)
,ROWS_INSERTED NUMERIC(10)
,ROWS_UPDATED NUMERIC(10)
,ROWS_DELETED NUMERIC(10)
,ROWS_REJECTED NUMERIC(10)
,ROWS_TREATED NUMERIC(10)
,NUM_TRANSFER NUMERIC(15)
)

;
```

Figure 21 : Script de création AUDIT_DETAILS

	NUM_DETAIL	TABLE_NAME	ROWS_INSERTED	ROWS_UPDATED	ROWS_DELETED	ROWS_REJECTED
1	1	OUTLET_LOOKUP	3	0	0	0
2	2	SHOP_FACTS	4	0	0	0
3	3	ARTICLE_LOOKUP	3	0	0	0
4	4	ARTICLE_COLOR_LOOKUP	3	0	0	0
5	5	CALENDAR_YEAR_LOOKUP	2	0	0	0

Figure 22 : Table AUDIT_DETAILS

- La table *AUDIT_ERROR* englobe toutes les erreurs qui ont eu lieu pendant le transfert et indique la source du problème. NUM_TRANSFER fait référence au numéro du transfert correspondant dans *AUDIT_TRACE*.

```
CREATE TABLE AUDIT_ERRORS
(
NUM_ERROR NUMERIC(10) NOT NULL IDENTITY(1,1)
,TABLE_NAME VARCHAR(25)
,PK_VALUE NUMERIC(10)
,SRC_PROBLEM VARCHAR(100)
,NUM_TRANSFER NUMERIC(15)
)

;
```

Figure 23 : Script de création AUDIT_ERRORS

c. Cas particulier de suppression :

Dans un modèle en étoile, la suppression est très compliquée et peut entrainer des incohérences. On prend l'exemple de la suppression d'un magasin, sachant que la table des faits peut contenir une référence sur ce dernier. Donc plusieurs solutions sont possibles dont la suppression des enregistrements faisant référence à ce magasin dans la table de faits est une solution. Dans le cas de notre projet, il est important de garder un historique des ventes même si le magasin est supprimé, d'où la mise en place d'un mécanisme de suppression logique.

Sachant que le fonctionnement pour les trois autres dimensions est identique, le fonctionne de la suppression logique pour la table OUTLET_LOOKUP est le suivant :

- L'ajout d'une nouvelle colonne dans la table OUTLET_LOOKUP, nommée « ISACTIVE ». Elle prend la valeur 1 en cas d'existence du magasin et 0 si le magasin est déjà supprimé.

- Lors de la suppression d'un magasin qui possède des enregistrements associés dans la table des faits, la valeur de cette colonne devient 0 afin de conserver l'historique des ventes.

- La valeur par défaut de cette colonne est 1.

- En utilisant la transformation « MERGE JOIN », on vérifie la suppression logique ou physique au niveau du package, sur des données issues de différentes sources.

d. Cas particulier de la table de faits

La table SHOP_FACTS, étant la table de faits est gérer différemment des dimensions. On commence par l'insertion des données dans les dimensions avant l'insertion dans la table de faits alors que pour la suppression doit se faire avant les modifications dans les dimensions pour éviter les erreurs de violation de contrainte d'intégrité. Pour les modifications est un cas particulier qui sera assimilée à une suppression dans un premier temps puis à une insertion.

e. Fonctionnement globale du package

Les fonctionnalités de ce package sont :

- Initialisation de l'audit selon la table :
    - AUDIT_TRACE : génération d'un nouveau numéro de transfert et l'insertion de la date et l'heure de début.
    - AUDIT_DETAILS : initialisé à « 0 » des statistiques pour chaque table.
- Gestion des modifications ainsi que les suppressions dans la table SHOP_FACTS.
- Transfert des données vers les dimensions.
- Gestion des insertions pour la table SHOP_FACTS.
- Génération automatique du code erreur pour le transfert.
- Suppression des données dans le schéma EMODE_INC.
- Finaliser l'audit par l'ajout de la date et de l'heure indiquant la fin du transfert.

Pour décrire le processus global de ce package, dans un premier temps, plusieurs vérifications manuelles au niveau des données pour garder seulement les données saines pour les traiter avec des requêtes SQL. Pour ce traitement on imagine tous les scénarios possibles comme le type de suppression s'il doit être logique ou physique puisqu'on n'utilise pas les outils de gestion d'erreurs proposé par SSIS. Ainsi la gestion du transfert des données pour la table de faits et les dimensions est traiter selon la même approche.

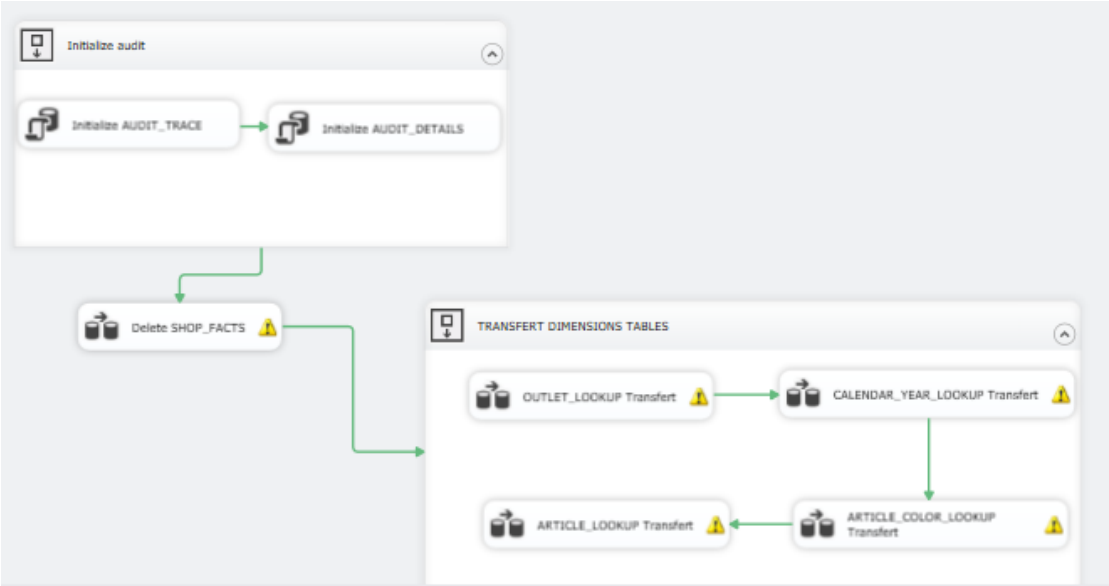


Figure 24 : Vue Globale package 2 - partie 1

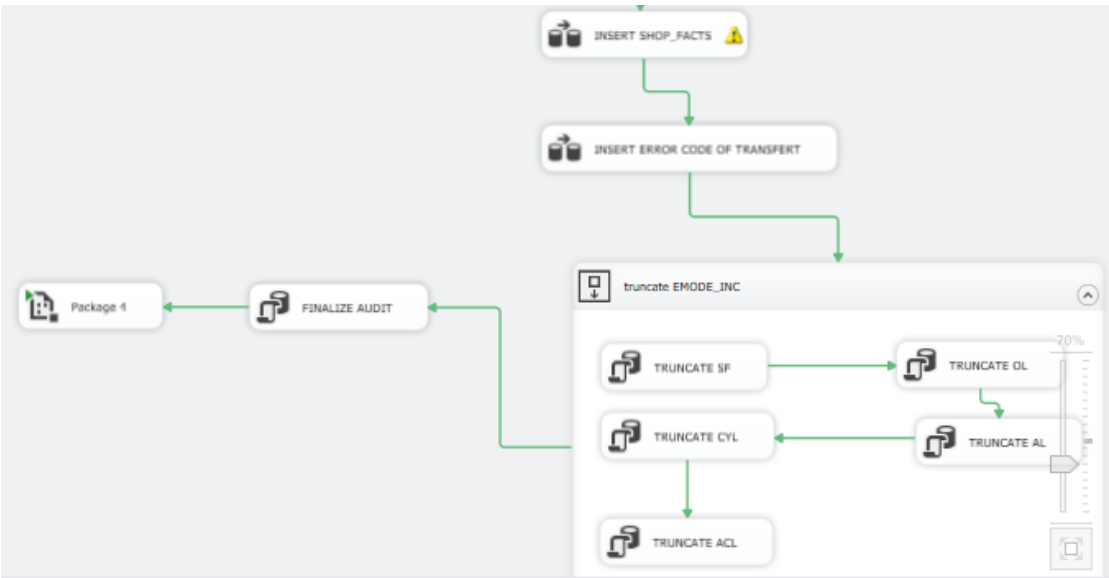


Figure 25 : Vue Globale package 2 - partie 2

Gestion des erreurs

Les types d'erreurs traitées au niveau de ce package sont :

- L'existence de la même clé primaire pour une insertion
- Clé primaire avec valeur « null »
- L'inexistence de la clé primaire pas pour une suppression ou une mise à jour
- L'inexistence des clés étrangères n'existe pas dans une dimension lors d'une insertion dans SHOP_FACTS

Le traitement des différentes erreurs est pareil : On commence tout d'abord par une vérification avec les transformations de type « LOOKUP » ou « MERGE JOIN » selon la source des données. Ensuite, les données qui posent un problème sont redirigées vers la table de rejet correspondante, simultanément on alimente la table AUDIT_STATS par le nombre de lignes rejetées pour une insertion qu'on a compté. Finalement, la table AUDIT_ERROR aura deux nouvelles colonnes : « SRC_PROBLEM » et « TABLE_NAME » pour alimenter le flux d'erreur avec plus de détails.

Gestion des statistiques

A l'aide de la transformation « AGGREGATE », on compte le nombre de lignes traitées et enregistrer dans le flux. Ainsi on alimente la table AUDIT_STATS au fil du traitement.

3. Package 3

Ce package a les même fonctionnalités ainsi que le même enchainement d'étapes package 2, qui réside en la gestion du transfert incrémental des données ainsi que d'assurer l'audit pour ce dernier. Pour ce package, on mettra en fonction les mécanismes d'audits proposés par « Integration Services ». Ce qui enchainera un changement dans le processus au niveau du Data Flow.

Contrairement au package précédent, où on devait vérifier et filtrer les données avant de commencer le traitement, le package 3 et grâce aux outils d'audits proposés on pourra passer directement à l'étape du traitement avec les requêtes SQL sans rencontrer des erreurs d'incohérence de données. C'est ainsi que les lignes problématique serviront pour alimenter les tables d'audit.

a. Les fonctionnalités d'Integration Services

Les outils d'audit que Integration Services met à disposition sont :

- La transformation « AUDIT », au sein d'un Data Flow, qui permet d'intégrer des données au enregistrements tel que : Exécution instance GUID, Execution

start time, Machine name, Package ID, Package name, Task ID, Task name, User name et Version ID.

- Les « Events Handlers » permet la gestion des évènements de type OnError, OnInformation, OnTaskFailed ou OnWarning… Ainsi de capturer des évènements lors de l'exécution d'un package.

- La possibilité de configuration automatique des logs au niveau des packages SSIS. On a l'exemple d'insertion automatiquement d'un certain nombre d'informations qu'on peut définir dans une table système crée par Integration Services ou dans les journaux d'évènements Windows. On peut paramétrer le niveau de détail et ainsi que ces types sont OnError, OnInformation, OnTaskFailed ou OnWarning par exemple.

- La possibilité de gestion automatiquement des erreurs à la sortie des transformations comme « OLE DB COMMAND », tel qu'ignorer ou rediriger les enregistrements problématiques. Cette transformation est caractérisée par une flèche rouge en sortie de transformation.

b. Mise en place de l'audit

Afin d'assurer un audit automatique du transfert et dans le cadre du projet, nous avons mise en place trois outils d'audit SSIS. Ainsi, nous avons créé deux nouvelles tables d'audit pour distinguer les deux approches d'audit.

La première table AUDIT_AUTO_TRACE contiendra des informations globales sur le transfert (numéro de transfert unique, date de début, date de fin, nom du package, nom de la machine et l'utilisateur qui exécute le package, code erreur, nombre de lignes traitées et nombre de lignes rejetées).

```
-- AUDIT_AUTO_TRACE
CREATE TABLE AUDIT_AUTO_TRACE(
NUM_TRANSFER NUMERIC(15) IDENTITY(1,1) NOT NULL,
START_TIME DATETIME,
END_TIME DATETIME,
PACKAGE_NAME VARCHAR(32),
MACHINE_NAME VARCHAR(32),
USER_NAME VARCHAR(32),
ERROR_CODE NUMERIC(1),
ROWS_TREATED NUMERIC(10),
ROWS_REJECTED NUMERIC(10)
);
ALTER TABLE AUDIT_AUTO_TRACE

ADD CONSTRAINT PK_AUDIT_AUTO_TRACE PRIMARY KEY (NUM_TRANSFER);
```

Figure 26 : Script de création AUDIT_AUTO_TRACE

La deuxième table AUDIT_AUTO_ERROR, contenaient (code erreur, description de l'erreur, la valeur de la clé primaire, nom de la table et le numéro du transfert concerné).

	NUM_ERROR	ERROR_CODE	ERROR_DESCRIPTION	PK_VALUE	TABLE_NAME	NUM_TRANSF
1	1	-1071607699	Conversion failed because the data value overflo...	263	CALENDAR_YEAR_LOOKUP	2
2	2	-1071607699	Conversion failed because the data value overflo...	264	CALENDAR_YEAR_LOOKUP	2
3	3	-1071607696	The data value violated the integrity constraints fo...	89172	SHOP_FACTS	2
4	4	-1071607696	The data value violated the integrity constraints fo...	89173	SHOP_FACTS	2
5	5	-1071607696	The data value violated the integrity constraints fo...	89174	SHOP_FACTS	2
6	6	-1071607696	The data value violated the integrity constraints fo...	89175	SHOP_FACTS	2

Figure 29 : Table AUDIT_AUTO_ERRORS

Transformation « AUDIT »

La table AUDIT_AUTO_TRACE sera alimenter par la transformation « AUDIT

	NU...	START_TIME	END_TIME	PACKAGE_N...	MACHINE_...	USER_NAME	ERROR_CODE	ROWS_TREATED
1	1	2019-01-03 00:59:00.000	2019-01-03 00:59:07.333	Package 3	PC-GI-2012...	PC-GI-2012-...	0	0
2	2	2019-01-03 01:31:17.000	2019-01-03 01:31:24.757	Package 3	PC-GI-2012...	PC-GI-2012-...	1	10
3	3	2019-01-03 01:48:49.000	2019-01-03 01:55:16.740	Package 3	PC-GI-2012...	PC-GI-2012-...	0	6

Figure 28 : Table AUDIT_AUTO_TRACE

», qui est utiliser pour initialiser l'audit au début du package, ainsi elle va servir pour alimenter la table AUDIT_AUTO_TRACE avec les paramètres suivants « Execution start time », « Package Name », « User Name » et « Machine Name ».

```
-- AUDIT_AUTO_ERRORS
CREATE TABLE AUDIT_AUTO_ERRORS
(
NUM_ERROR NUMERIC(10) NOT NULL IDENTITY(1,1)
,ERROR_CODE NUMERIC(10)
,ERROR_DESCRIPTION VARCHAR(255)
,PK_VALUE NUMERIC(10)
,TABLE_NAME VARCHAR(25)
,NUM_TRANSFER NUMERIC(15)
)
;
ALTER TABLE AUDIT_AUTO_ERRORS
ADD CONSTRAINT PK_AUDIT_AUTO_ERRORS PRIMARY KEY (NUM_ERROR);
ALTER TABLE AUDIT_AUTO_ERRORS
ADD CONSTRAINT FK_AUDIT_AUTO_ERRORS_NT
FOREIGN KEY (NUM_TRANSFER)

REFERENCES AUDIT_AUTO_TRACE(NUM_TRANSFER);
```

Figure 27 : Script de création AUDIT_AUTO_ERRORS

Tout au long du package, nous avons la possibilité d'utiliser cette transformation pour les données traités ou rejetés. Mais, on a choisi l'usage d'autres transformations proposées par Integration Services.

À divers emplacements dans le package, l'ajout du paramètre « Task Name » était possible, mais on l'a estimé utile seulement pour le débogage.

Gestion des erreurs des transformations

Pour la gestion des lignes qui ne respectent pas les contraintes d'intégrité, nous avons utilisé un mécanisme intégré à certaines transformations. Ce mécanisme permet d'ajouter deux colonnes (ErrorCode et ErrorColumn) afin de traiter les lignes qui ne respectent pas les contraintes d'intégrité.

En cours du projet, la table AUDIT_AUTO_ERROR et les tables de rejet, seront alimenter par les lignes redirigés avec une erreur.

À l'aide d'un script, il est possible d'avoir un descriptif associé pour le code d'erreur puisque ce dernier n'est pas explicite.

En sortie du script, on aura une nouvelle colonne ajoutée à la table : ErrorDescprion

Ce mécanisme réduit la précision de l'audit, tel que dans le cas d'une suppression d'un enregistrement non existant, aucune erreur n'est levée. Pourtant, il est simple a mettre en place et allège fortement la conception du package.

SSIS Logging

Ce mécanisme est mis en œuvre lors de notre projet pour tester quelques fonctionnalités, ainsi il permet de tracer un très grand nombre d'informations de manière automatique. Afin d'alimenter une table système créée automatiquement nous avons créé un log portant sur les évènements de type OnError et OnTaskFailed.

Transformation « ROW COUNT »

Contrairement au package 2, où nous avons utilisé la transformation « AGGREGATE » afin de compter les lignes traitées et rejetées et pour tester le plus de fonctionnalités d'Integration Services, on a utilisé la transformation « ROW COUNT » dans ce package dans le but de fournir des statistiques sur le nombre d'enregistrements traités et rejetés.

Cette transformation a pour but d'alimenter une variable créée manuellement au préalable. Cette variable à une portée modifiable, tel qu'on peut créer une variable utilisable uniquement pour un Data Flow ou tout un package.

Vers la fin de ce package, ces différentes variables seront utilisées pour alimenter la table AUDIT_AUTO_TRACE.

Name	Scope	Data type	Value
RowR	Package 3	Int32	0
RowRejected	Package 3	Int32	0
RowT	Package 3	Int32	0
RowTreated	Package 3	Int32	0

Figure 30 : Variables Row count

c. Fonctionnement globale du package

Les fonctionnalités de ce package sont :

- Initialisation de l'audit dans la table AUDIT_AUTO_TRACE génération d'un nouveau numéro de transfert, l'insertion de la date et l'heure de début, nom de package et de nom d'utilisateur tous ces informations sont générées par la transformation AUDIT.
- Gestion des modifications ainsi que les suppressions dans la table SHOP_FACTS.
- Transfert des données vers les dimensions.
- Gestion des insertions pour la table SHOP_FACTS.
- Génération automatique du code erreur pour le transfert.
- Suppression des données dans le schéma EMODE_INC.
- Finaliser l'audit par l'ajout de la date et de l'heure indiquant la fin du transfert.

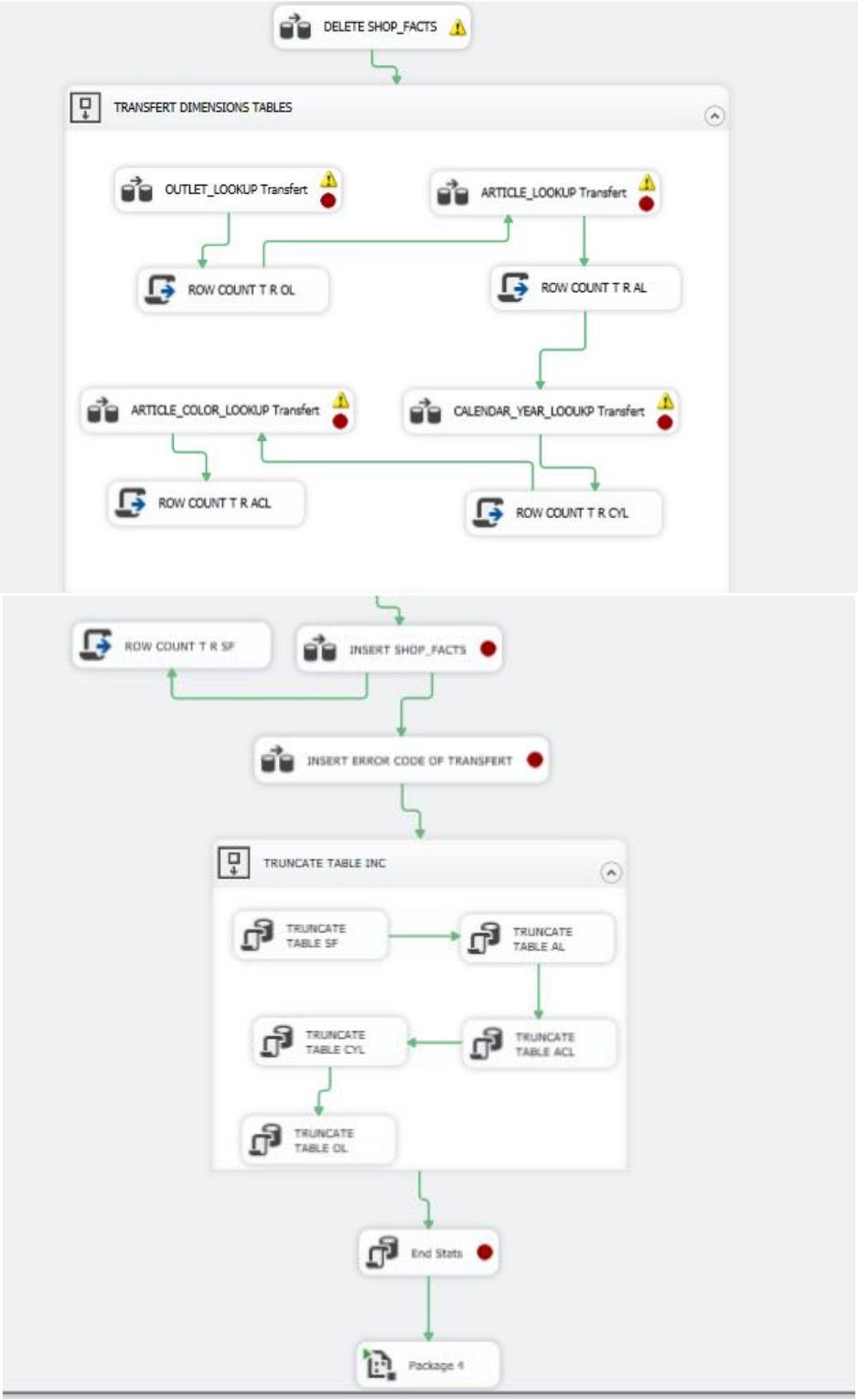


Figure 31 : Vue Globale package 3

4. Package 4

L'objectif de ce dernier package est l'alimentation des deux tables d'agrégation, qu'on a créé au préalable dans le schéma EMODE de SQL Server.

a. Fonctionnement global

Le processus global est très basique, nous commençons par vider les tables d'agrégations ensuite les alimenter avec les nouvelles données.

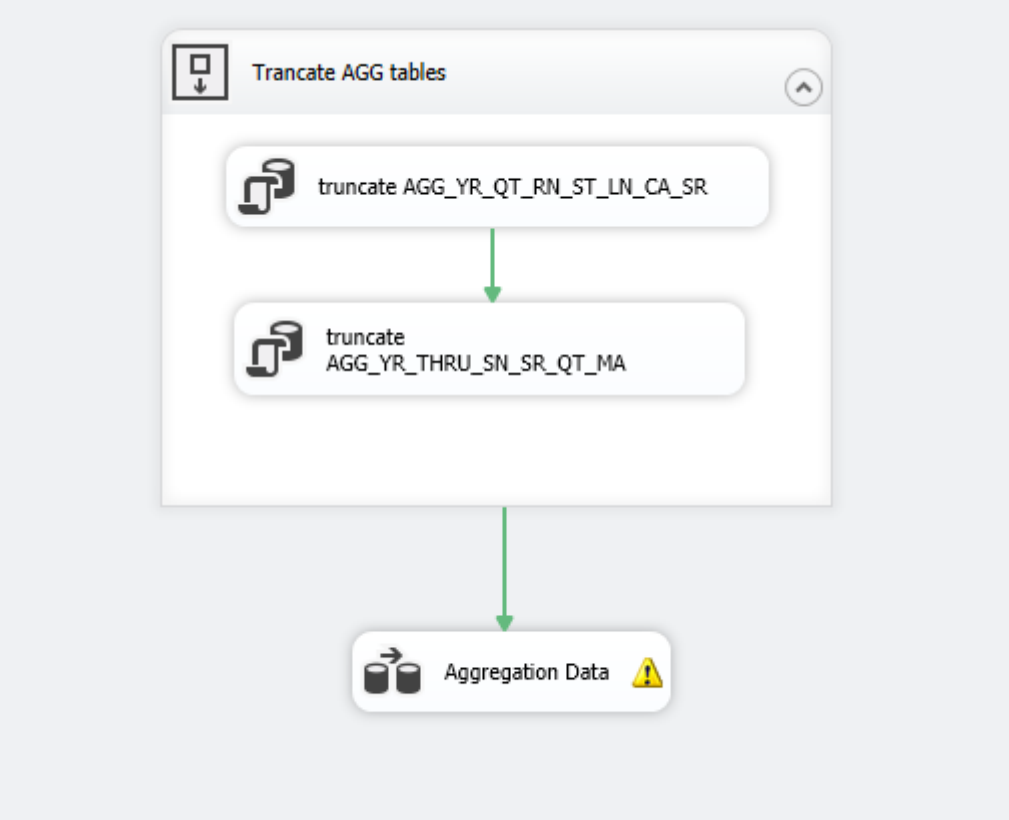


Figure 32 : Vue global package 4

b. Détail data flow

L'organisation du Data Flow est de la manière suivante :

- Déterminer les données nécessaires pour alimenter les deux tables d'agrégations.
- L'usage de la transformation « AGRREGATE » qui assure l'efficacité et la simplicité dans ce Data Flow.
- Etape finale, l'insertion des données dans les tables d'agrégations.

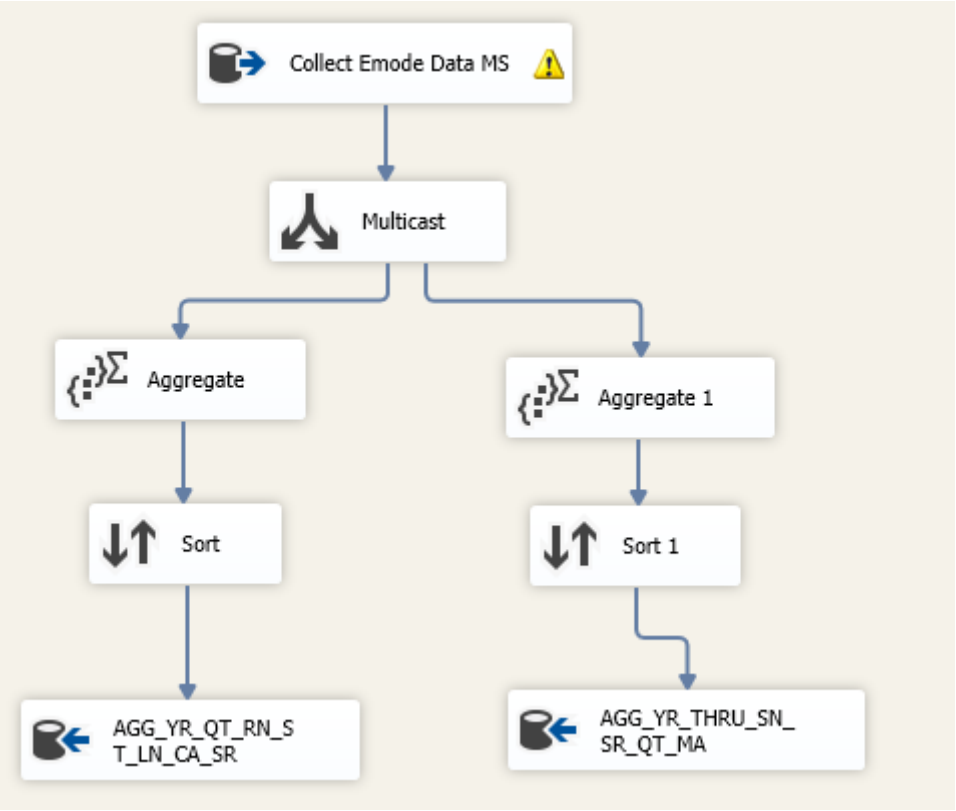


Figure 33 : Data Flow du package 4

c. Modification des autres packages

Integration Services permet l'appelle d'un package depuis un autre package grâce à la transformation « Execute Package Task ».

On fait appeler ce package depuis les trois packages précédents. Tel qu'après chaque exécution de l'un des packages, il faut générer des nouvelles données agrégées puisque la modification des données est très probable.

D'où on a modifié les trois packages précédents afin qu'on puisse exécuter ce dernier package juste avant la fin de leur exécution.

5. Test ETL

La phase de test se décompose en deux parties :

- Les tables sont vides dans la base de destination et nous effectuons un chargement initial des données à l'aide du premier package.
- Par la suite, des données sont insérées pour tester le transfert incrémental.

a. Exécution du package 1

Fin de l'exécution du package 1

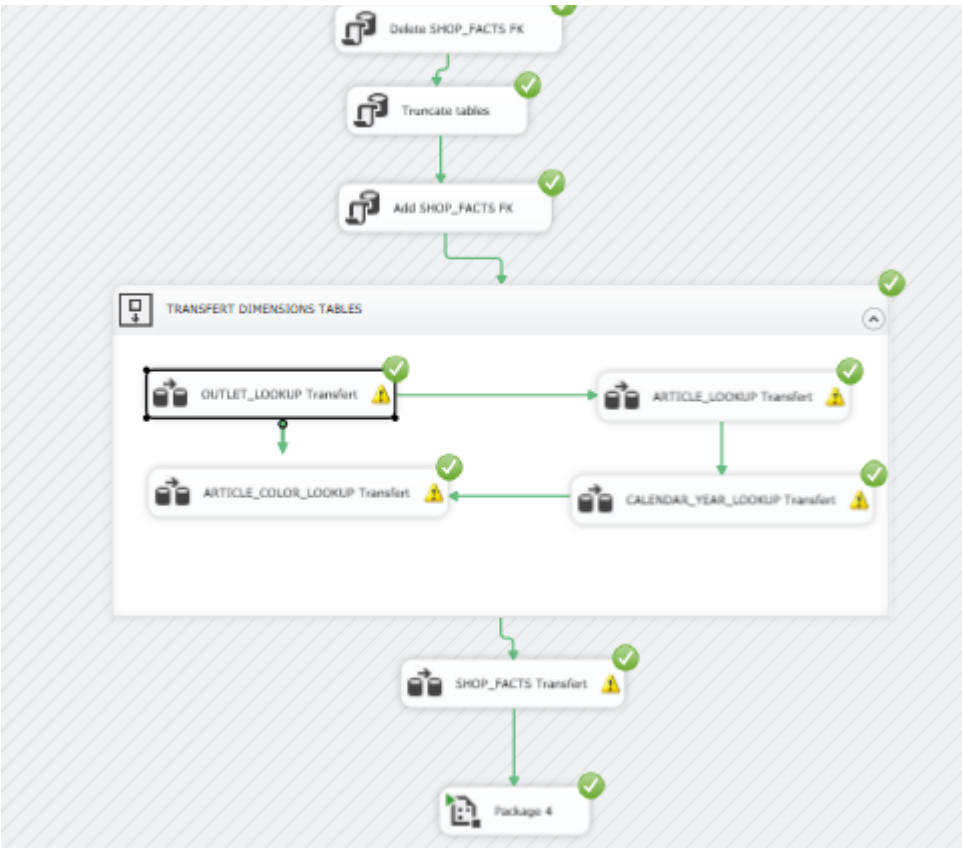


Figure 34 : Exécution du package 1

L'exécution de tous les composants est réalisée avec succès, ce qui est vérifié par la couleur verte.

Extrait Data Flow « OL Transfer »

Nous pouvons facilement visualiser le nombre de lignes rejetées. Dans ce cas, on a l'ajout de 13 lignes et aucune ligne rejeté, d'où on déduit que toutes les données sont cohérentes.

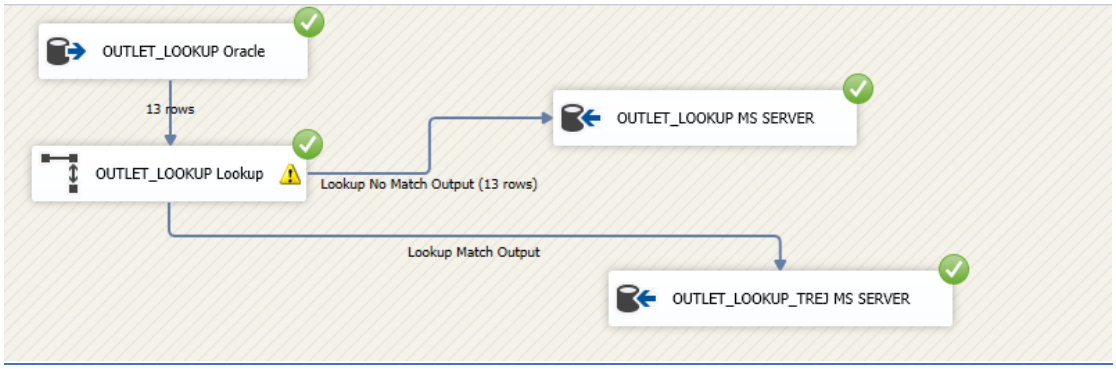


Figure 35 : Exécution du transfert de OUTLET_LOOKUP - Package 1

b. Exécution du package 2

Pour cette phase de test, nous avons inséré deux nouvelles semaines, trois nouveaux magasins, trois nouveaux produits et quatre ventes. Nous avons exécuté le package 2 et 3 pour faire ces tests.

Aperçu d'une table du schéma EMODE_INC pour CYL

	WEEK_KEY	WEEK_IN_YEAR	YEAR	FISCAL_PERIOD	YEAR_WEEK	QUARTER	MONTH_NAME	MONTH	HOLIDAY_FLAG	OPERATION_TYPE
1	263	1	2018	FY18	2018/1	1	January	1	y	INSERT
2	264	2	2018	FY18	2018/2	1	January	1	n	INSERT

Figure 36 : Ajout des lignes sur EMODE_INC

Fin de l'exécution du package 2

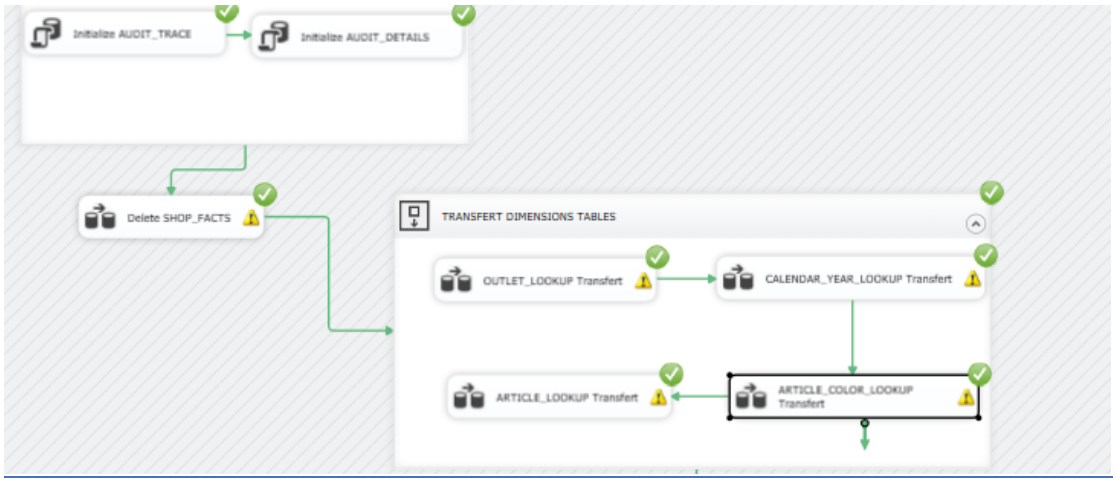


Figure 37 : Exécution du package 2

Extrait du Data Flow « Transfer CYL »

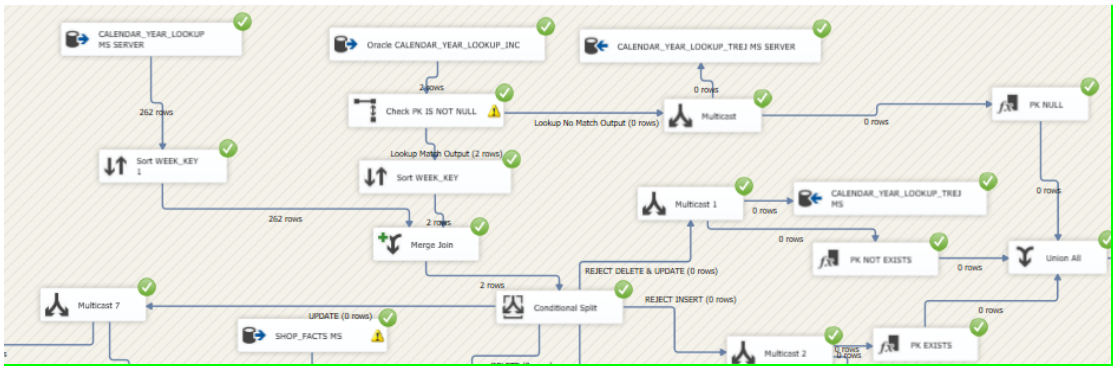


Figure 38 : Exécution du Data Flow CYL

Nous pouvons constater le transfert des deux nouvelles semaines a bien été fait vers la base de destination.

Aperçu d'une table dans la base de destination

263	263	1	2018	FY18	2018/1	1	January	1	y	1
264	264	2	2018	FY18	2018/2	1	January	1	n	1

Figure 39 : Ajout des lignes sur SQL SERVER

Et voilà ce qui prouve que la présence des deux nouvelles semaines dans la base de destination.

c. Exécution du package 3

Pour l'exécution du package 3 c'est comme le package 3, afin de le testé nous avons modifié les ligne que nous avons insérée avec le package 2.

Fin de l'exécution du package 3

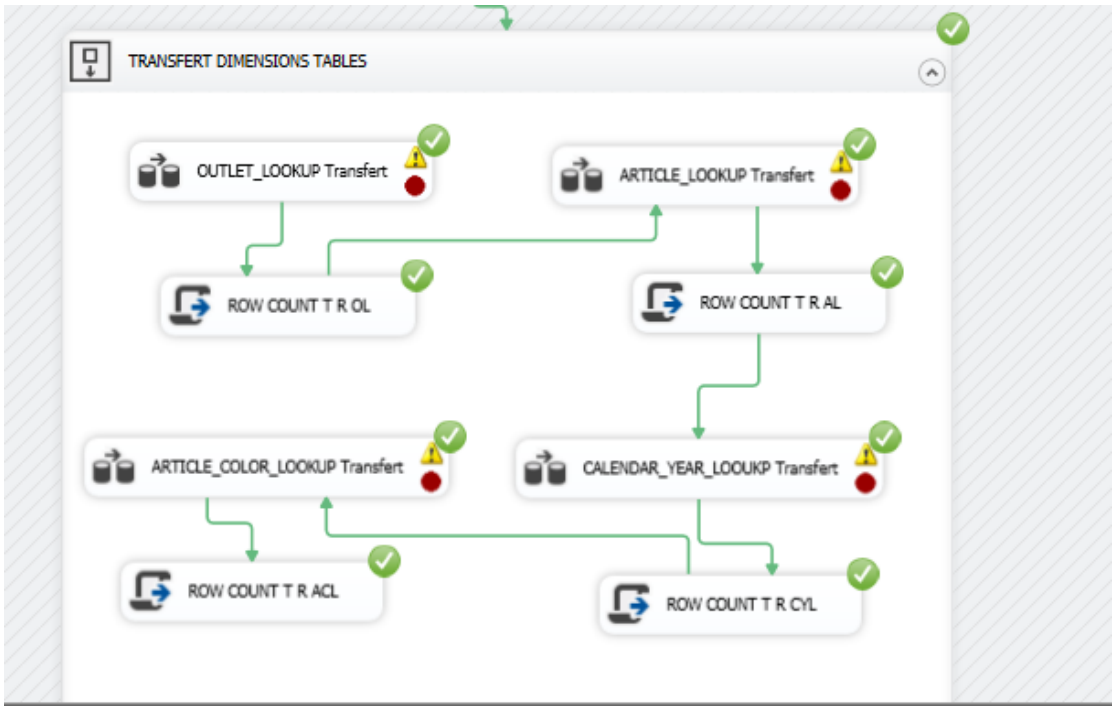


Figure 40 : Exécution du package 3

Data Flow « Transfer SF »

Figure 41 : Exécution Data Flow SF

d. Exécution du package 4

Comme ce que nous avons précisé avant l'exécution du package 4 ce fait après l'exciton des autres packages et permet d'agrégé les données dans les tables d'agrégations.

Fin de l'exécution du package 4

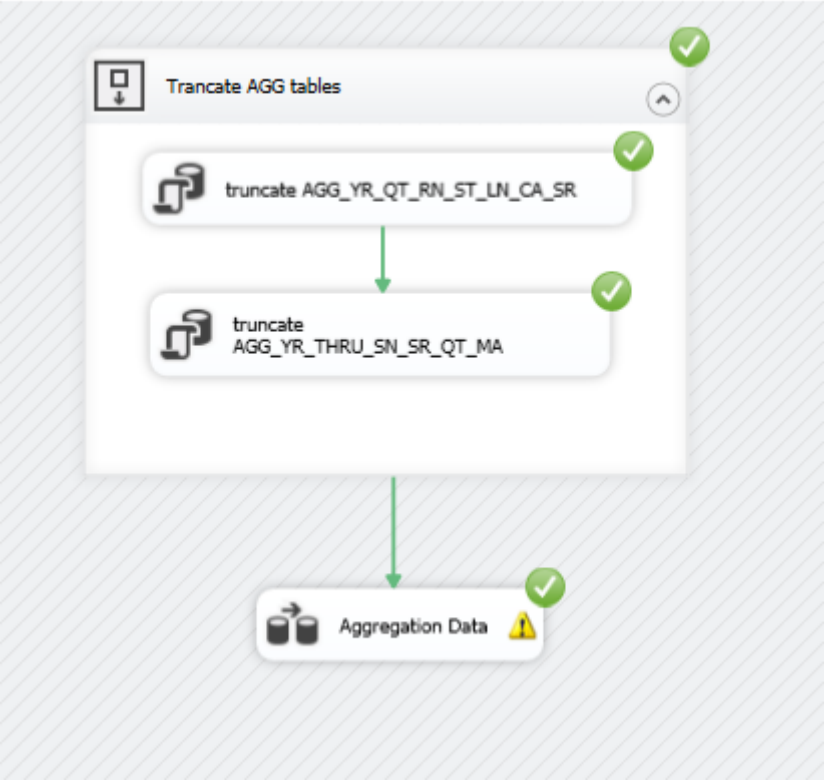


Figure 42 : Exécution du package 4

6. Automatisation de l'exécution des packages

a. Exécution immédiate

Il existe la possibilité d'exécuter un package SSIS, tel qu'après avoir créé un package, nous pouvons « Build » le projet, sans passer par « SQL Server Business Intelligence Development Studio ». Ensuite on s'aperçoit, dans un répertoire paramétrable, la création automatique d'un nombre de fichiers .dtsx dont chacun correspond à un package. Dans les options. A chaque package correspond un fichier .dtsx.

L'exécution du package est très simple qui se résume en 2 étapes, un double-clique sur le ficher .dtsx puis en cliquer sur « Execute », ce qui est possible grâce à l' « Execute Package Utility »,

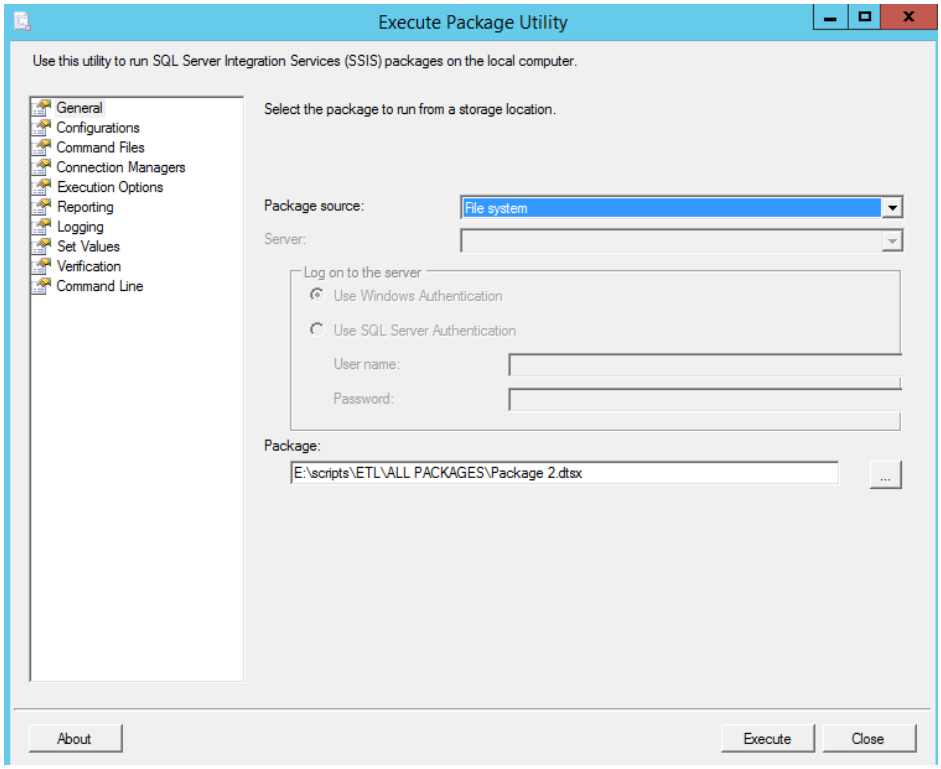


Figure 43 : ETL Exécution immédiate

b. Taches planifiées system

Dans le but d'automatiser l'exécution des packages, nous allons utiliser l'outil de planification des tâches fourni par Windows.

Nous avons la possibilité d'exécuter directement les fichiers .dtsx à travers l'assistant de paramétrage des tâches planifiées, mais pour planifier une exécution automatique durant la nuit par exemple, cela n'est pas optimisé. Ainsi, nous avons dédier un fichier de commande pour lancer le package et générer un fichier log pour avoir des traces sur le déroulement de la tâche.

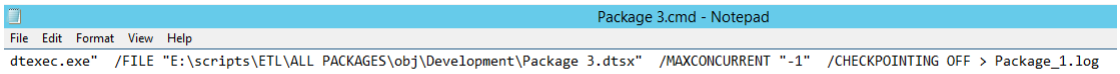


Figure 44 : Exécution du package 3 CMD

La configuration de la tâche planifiée est la suivante :

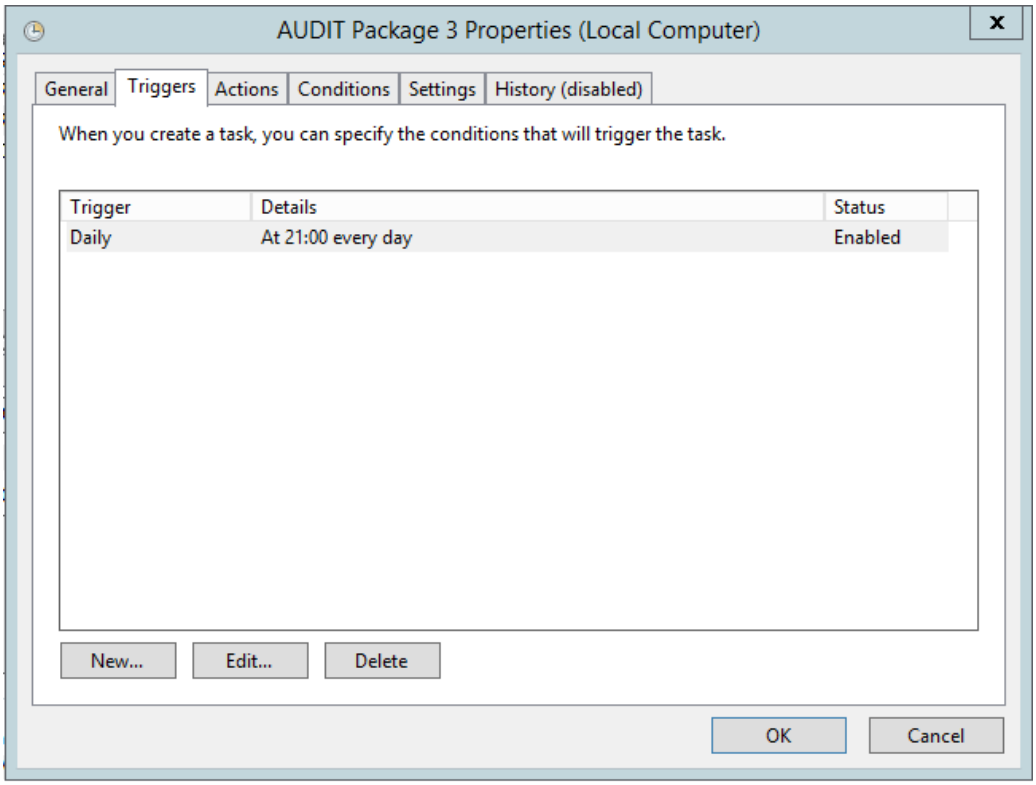


Figure 45 : Tâche planifiée Windows

c. SQL Server Agent

En utilisant SQL Server Agent, on peut exécuter un package SSIS. A travers un d'un service de SQL Server, accessible via « Microsoft SQL Server Management Studio » qui permet une exécution de travaux selon le besoin (automatique ou non).

Nous avons créé un Job Exécution ETL afin d'exécuté le Package 2 chaque jour à 20h30.

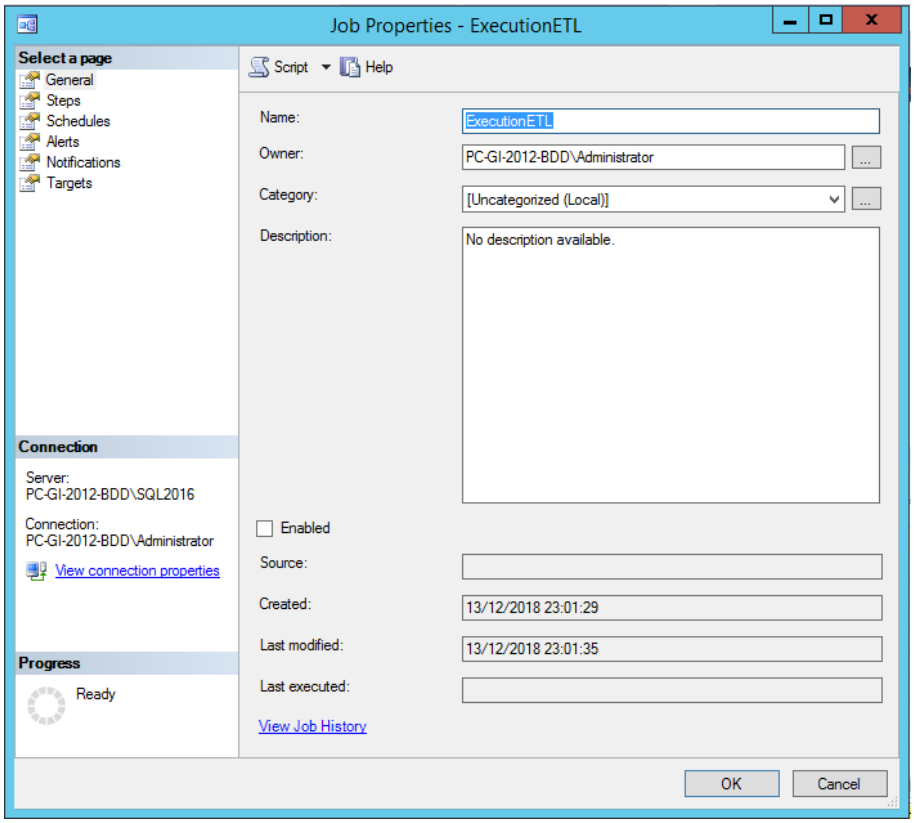


Figure 46 : Création du Job SQL Server Agent

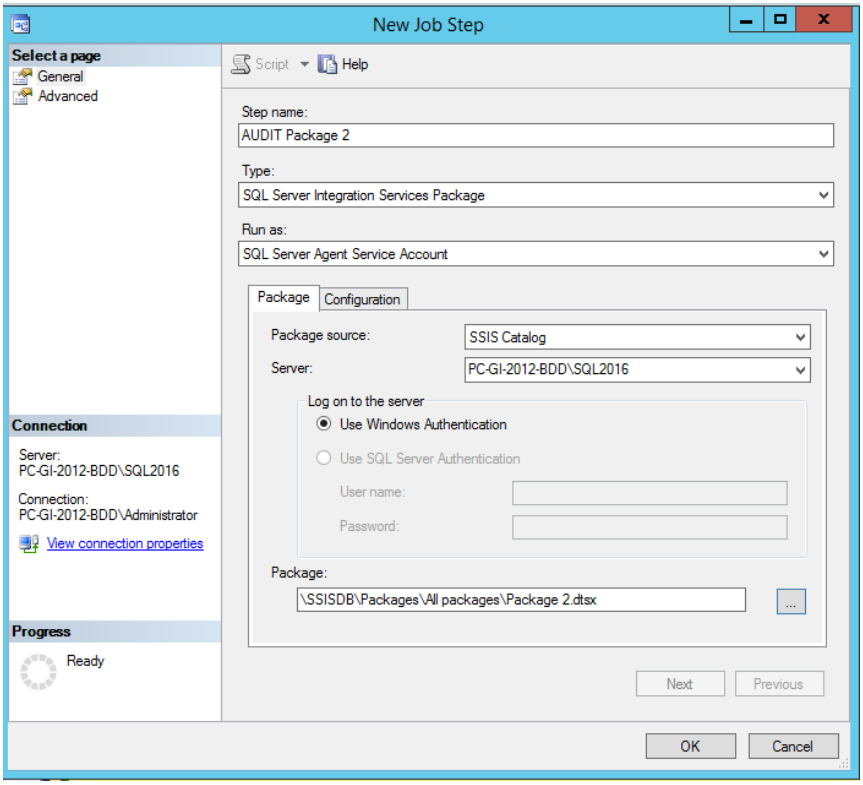


Figure 47 : Création du Job Audit Package 2

Le lancement du « Job » peut être immédiatement ou planifié selon notre choix grave au planificateur « Job Schedule »

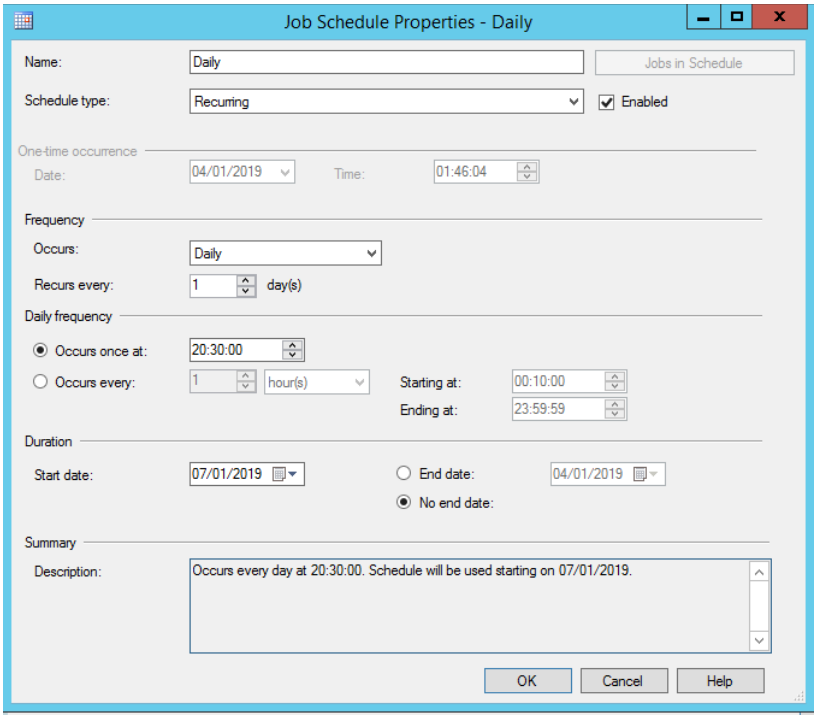


Figure 48 : Création du planificateur Job Schedule

Partie 2 : Optimisation du Data Warehouse

III. PARTITIONNEMENT DE LA TABLE DE FAITS

Le partitionnement d'une table est le fait de la découper horizontalement en sous-ensemble de taille plus optimal. La table devient donc distribuée entre plusieurs groupes de fichiers physiques. Ce system est totalement transparent pour l'utilisateur et le SGBD, qui se charge de distribuer les requêtes sur les partitions concernées.

Le partitionnement peut être bénéfique à plusieurs niveaux, surtout pour les tables comportant beaucoup de lignes. En effet, ce repartitionnement des données les rend plus facile à gérer e, terme de disponibilité et d'accessibilité. Si on utilise la stratégie « diviser pour régner » et on stocke des partitions sur différents disques durs dans un souci de continuité de service. Ce qui rend les données disponibles alors que d'autres sont inaccessibles à la suite d'une panne. Ce découpage garanti une amélioration des performances grâce aux traitements parallèles.

Dans le cadre de ce projet, la table SHOP_FACTS est susceptible d'être soumise à un partitionnement. Cette table contient environ 90000 lignes donc un partitionnement dans le but d'améliorer les performances n'est pas utile mais un groupement de données de manière logique peut être intéressant. Donc, on a choisi d'utiliser le groupement logique par année. De plus, dans le cadre du projet, le nombre de lignes par année est relativement important à savoir 17000 lignes pour 1999, 28000 lignes pour 2000 et 31000 pour 2001. Dans la table SHOP_FACTS, on pourra ainsi utiliser la colonne « WEEK_KEY » comme colonne de partitionnement. On va pouvoir prendre des plages de 52 « WEEK_KEY » pour définir une année. Etant donné qu'il n'y a pas de données entre 2001 et 2018 le découpage se fait de la manière suivante :

Intervalle des clés	Année
1 à 52	1997
53 à 104	1998
105 à 156	1999
157 à 209	2000
210 à 262	2001
263 à …	2018

1. Mise en œuvre du partitionnement

Afin de partitionner une table, le passage par plusieurs étapes est nécessaire.

a. Espaces de stockages

Le stockage des partitions sera sur différent espace de stockage appelé également « storage », selon notre choix. Chaque strage est associé à un fichier physique dont on peut contrôler la taille et la stratégie de croissance. Dans le projet, tous les fichiers physiques sont sur le même disque dur mais nous pouvons supposer que le stockage de chaque fichier sur un disque dur différent.

Le résultat de notre stratégie de partitionnement, sera donc 6 espaces de stockage.

- La création d'un « storage » par un script.

```
-- Storages
ALTER DATABASE EMODE
ADD FILEGROUP SF_FG_01;

ALTER DATABASE EMODE
ADD FILEGROUP SF_FG_02;
```

Figure 49 : Script de création du storage

- L'association du storage à un fichier physique.

```
-- Storage files

ALTER DATABASE EMODE
   ADD FILE (NAME = 'SF_FL_01',
             FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FL_01.ndf',
             SIZE = 1 GB,
             FILEGROWTH = 5 MB)
TO FILEGROUP SF_FG_01;

ALTER DATABASE EMODE
   ADD FILE (NAME = 'SF_FL_02',
             FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FL_02.ndf',
             SIZE = 1 GB,
             FILEGROWTH = 5 MB)
TO FILEGROUP SF_FG_02;
```

Figure 50 : Script d'association au fichier physique

b. Fonction de partition

On fait appeler une fonction de partition, qui va définir le mapping des lignes de la table en se basant sur les valeurs d'une colonne WEEK_KEY. Selon notre stratégie de partitionnement, les intervalles de valeur seront de 52 « WEEK_KEY ».

```
-- Partitioning function
CREATE PARTITION FUNCTION SF_PF (NUMERIC(3,0))
AS
  RANGE RIGHT

  FOR VALUES (53,105,157,210,263);
```

Figure 51 : Script de fonction de partition

c. Schéma de partition

Le schéma de partition mappe chaque partition spécifiée par la fonction de partition avec un groupe de fichier.

```
-- Partitioning schema
CREATE PARTITION SCHEME SF_PS
AS PARTITION SF_PF

TO (SF_FG_01,SF_FG_02,SF_FG_03,SF_FG_04,SF_FG_05,SF_FG_06);
```

Figure 52 : Schéma de partition

Le dernier « storage » SF_FG_06 va contenir toutes les données relatives depuis l'année 2018 jusqu'aux années suivantes.

d. Création de la table

Afin de prendre en compte le partitionnement à la création de la table, nous utilisons la requête suivante :

```
-- Table temp
CREATE TABLE SHOP_FACTS_TEMP(
ID NUMERIC (5,0) NOT NULL
, ARTICLE_CODE NUMERIC (6,0)
, COLOR_CODE NUMERIC (4,0)
, WEEK_KEY NUMERIC (3,0)
, SHOP_CODE NUMERIC (3,0)
, MARGIN NUMERIC (18,0)
, AMOUNT_SOLD NUMERIC (13,2)
, QUANTITY_SOLD NUMERIC (13,2)

)
```

Figure 53 : Script de création de table TEMP

e. Ajout des contraintes

Finalement, il fallait ajouter les contraintes d'intégrité sur la table SHOP_FACTS :

```sql
-- Add constraints
ALTER TABLE SHOP_FACTS
ADD CONSTRAINT PK_SHOP_FACTS PRIMARY KEY NONCLUSTERED (ID)
ON [PRIMARY];

ALTER TABLE SHOP_FACTS
ADD CONSTRAINT FK_SHOP_FACTS_ARTICLE_COLOR_LOOKUP
FOREIGN KEY (ARTICLE_CODE,COLOR_CODE)
REFERENCES ARTICLE_COLOR_LOOKUP(ARTICLE_CODE,COLOR_CODE);
```

Figure 54 : Script d'ajout de contrainte

Après le passage des données vers la table temporaire, nous les avons basculés vers la nouvelle table partitionnée.

IV. PROJET ANALYSIS SERVICES

SQL Server Analysis Services (SSAS), nous permet la mise en place de « Online Analytical Processing » ou OLAP. OLAP est un cube qui permet la représentation d'une base multidimensionnelle. Chaque dimension du cube correspond à une dimension d'analyse. Cette représentation abstraite de données numériques facilite l'exécution des requêtes, ainsi nous permet de fournir des indicateurs suivant des dimensions pour un système d'aide à la décision.

Dans le cadre du projet, nous avons créé un cube SSAS à partir de la base SQL EMODE Le cube qu'on a créé est multidimensionnel dont chaque dimension représente un axe d'analys, voilà nos trois axes qu'on a choisis :

- Dimension temporelle basée sur la date de vente
- Dimension géographique basée sur la position du magasin
- Dimension basée sur le type de produit vendu

1. Source de données

La base EMODE sur SQL Server représente notre source de donnée, ainsi, les tables SHOP_FACTS, CALENDAR_YEAR_LOOKUP, OUTLET_LOOKUP, ARTICLE_COLOR_LOOKUP et ARTICLE_LOOKUP vont alimenter le cube.

Le schéma suivant représente la source de données :

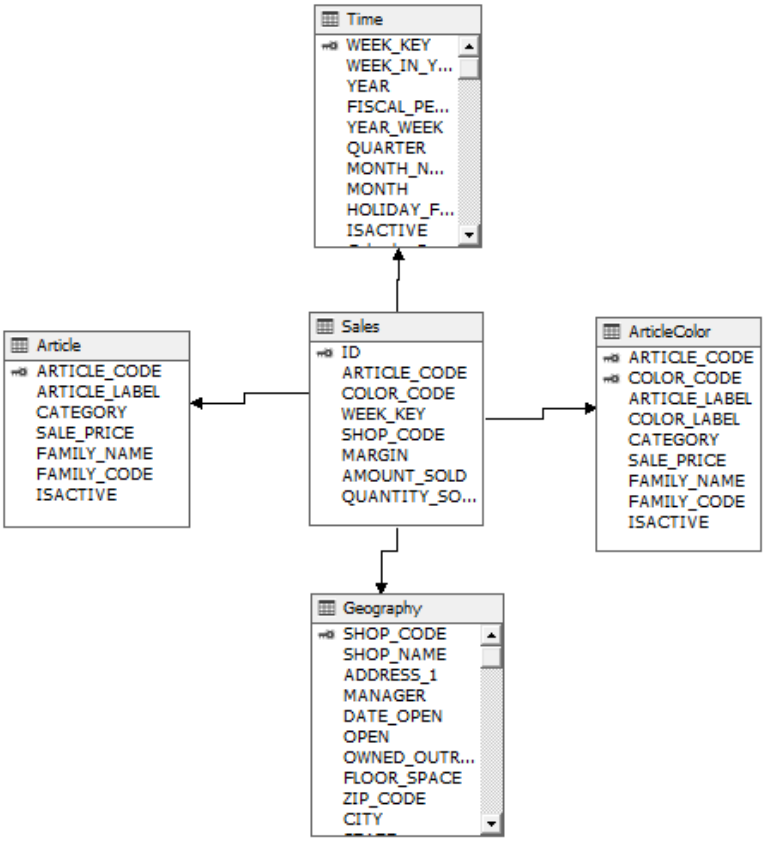


Figure 55 : Figure - Source de données cube OLAP

Afin de faciliter la lecture et la navigation dans le cube, les tables ont été renommées.

2. Calcul nommé

Nous avons créé des «Named Calculation» pour faciliter la lecture de certains attributs.

MONTH YEAR

Ce calcul nommé affiche : January 2001, le nom du mois suivi de l'année.

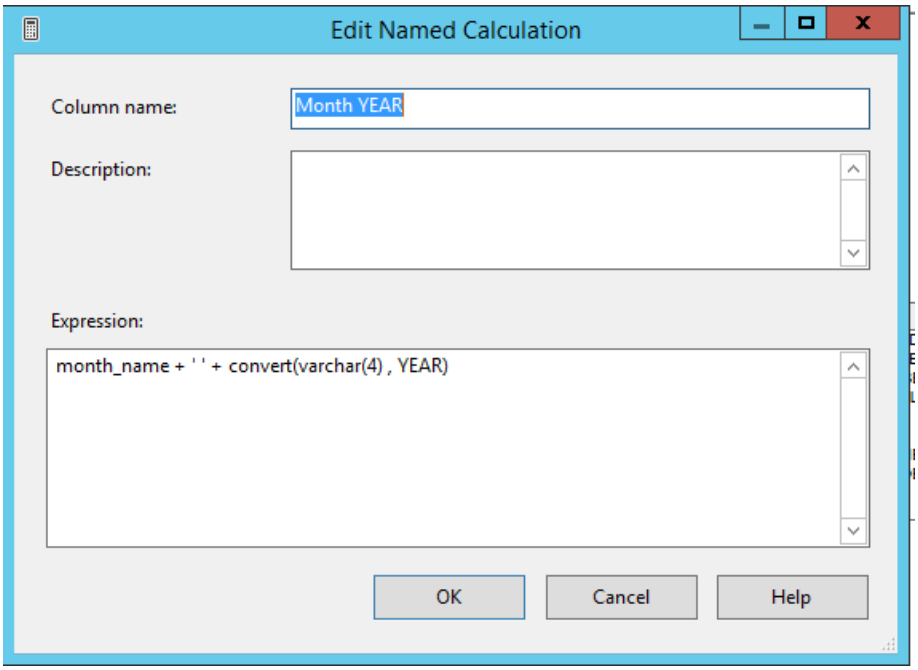


Figure 56 : Figure Propriété MONTH YEAR

WEEK

Ce calcul nommé affiche : 03 2001, la semaine sur deux caractères et l'année.

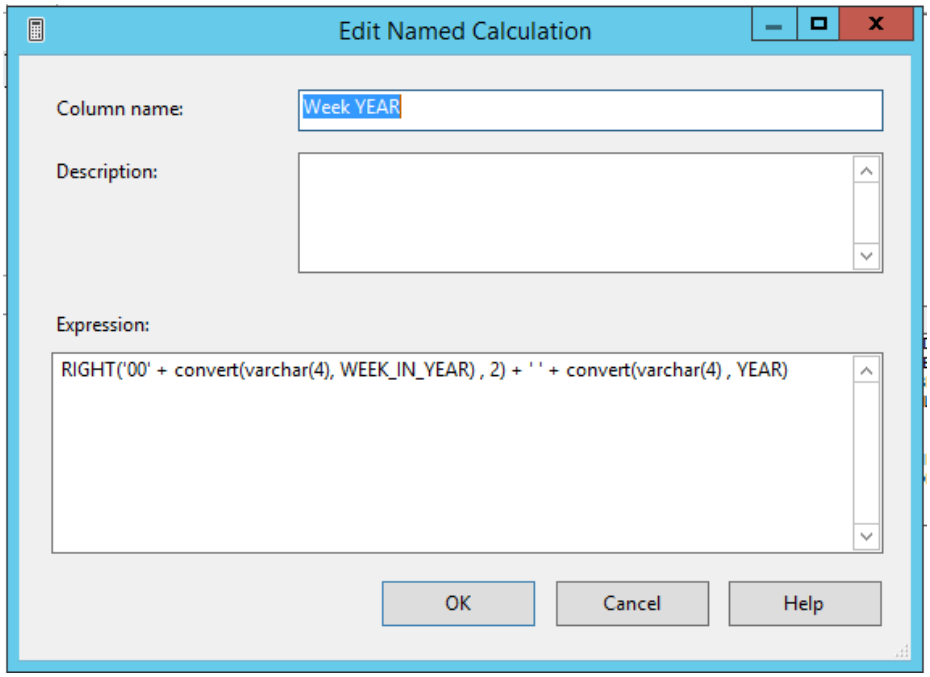


Figure 57 : Propriété WEEK

3. Dimensions

a. Temps

La table CALENDAR_YEAR_LOOKUP est la source principale des attributs de la dimension de temps, d'où on a retenu juste les données qui sont pertinentes pour notre projet. Voilà les colonnes que nous avons gardées : CALENDAR QUARTER, MONTH NAME, WEEK_KEY, WEEK, YEAR.

Ces attributs permettront d'avoir des indicateurs par semaine, par mois, par trimestre et par année.

On a mis en place une hiérarchie, pour faciliter l'exploration du cube.

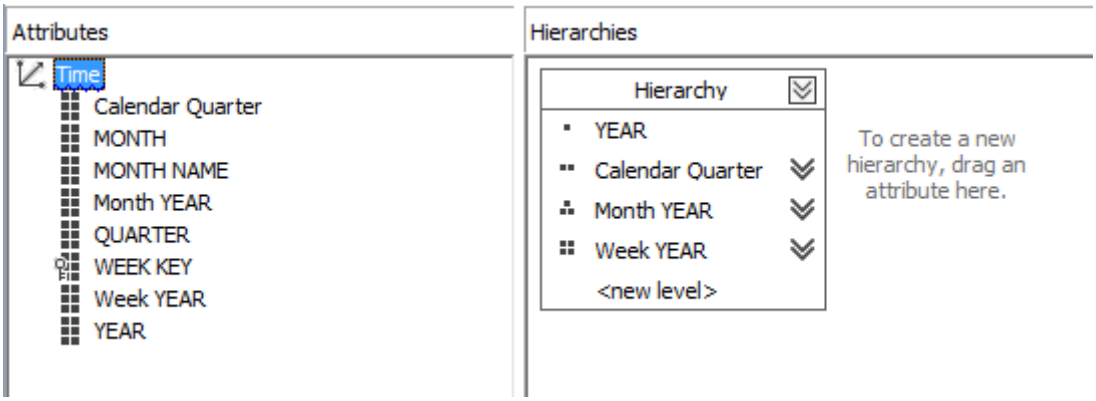


Figure 58 : Attributs et hiérarchie dimension "Time"

La navigation dans la dimension se présente de la manière suivante :

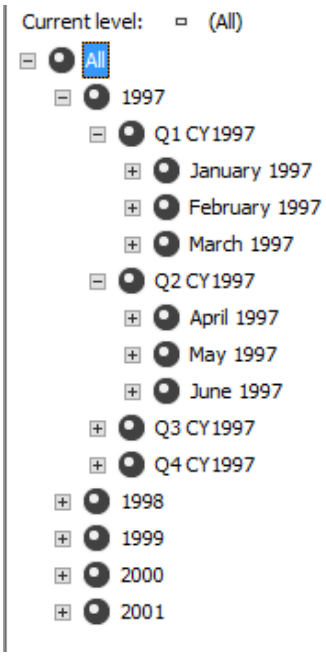


Figure 59 : Navigation dimension "Time"

b. Géographie

Les attributs de cette dimension permettent d'avoir des indicateurs par état, ville et magasin.

On a mis en place une hiérarchie, pour faciliter l'exploration du cube.

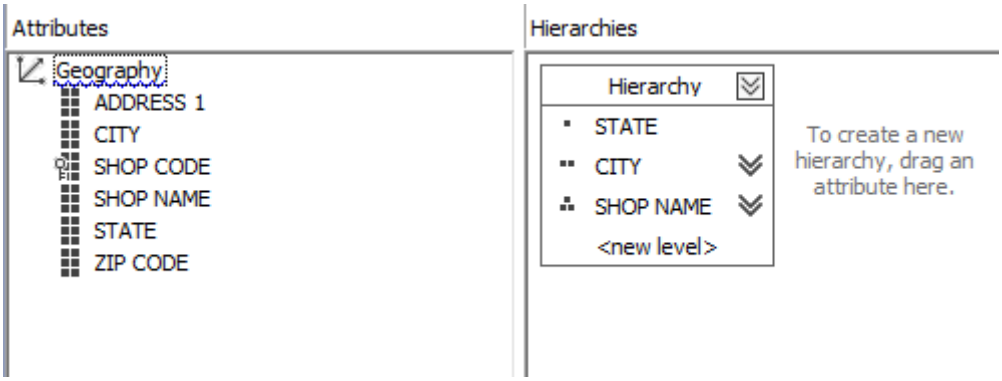


Figure 60 : Attribut et hiérarchie dimension "Geography"

La navigation se présente ainsi :

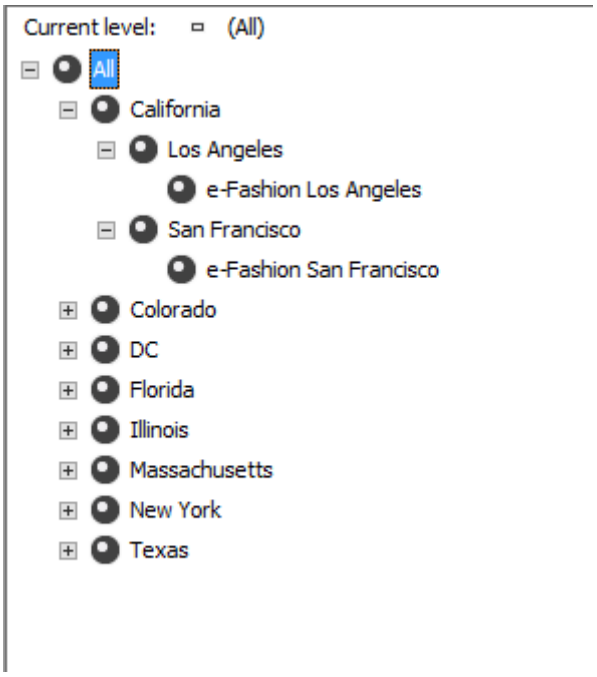


Figure 61 : Navigation dimension "Geography"

c. Article

Les attributs de cette dimension permettent d'avoir des indicateurs par famille d'article, par catégorie d'article et par article.

On a mis en place une hiérarchie, pour faciliter l'exploration du cube.

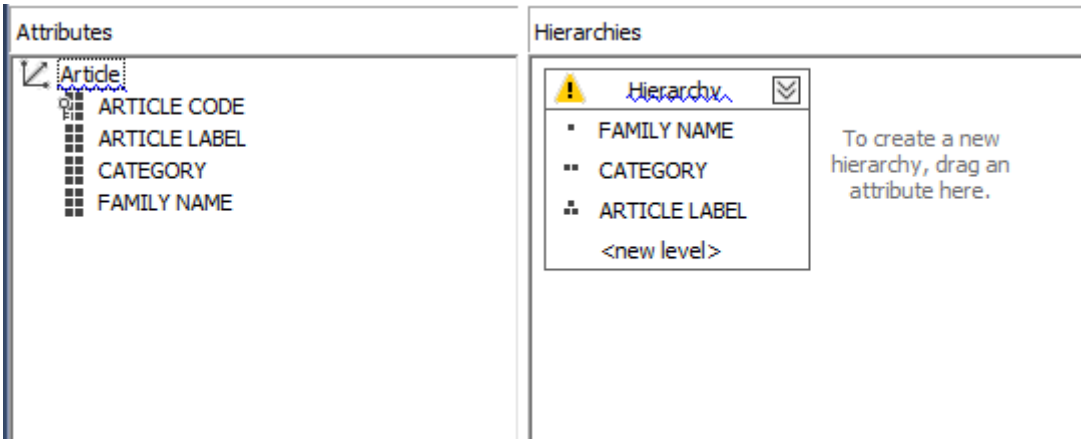


Figure 62 : Attributs et hiérarchie dimension "Article"

La navigation se présente ainsi :

Figure 63 : Navigation dimension "Article"

d. Article Couleur

Les attributs de cette dimension permettent d'avoir des indicateurs par famille d'article, par catégorie d'article, par article et par couleur.

On a mis en place une hiérarchie, pour faciliter l'exploration du cube.

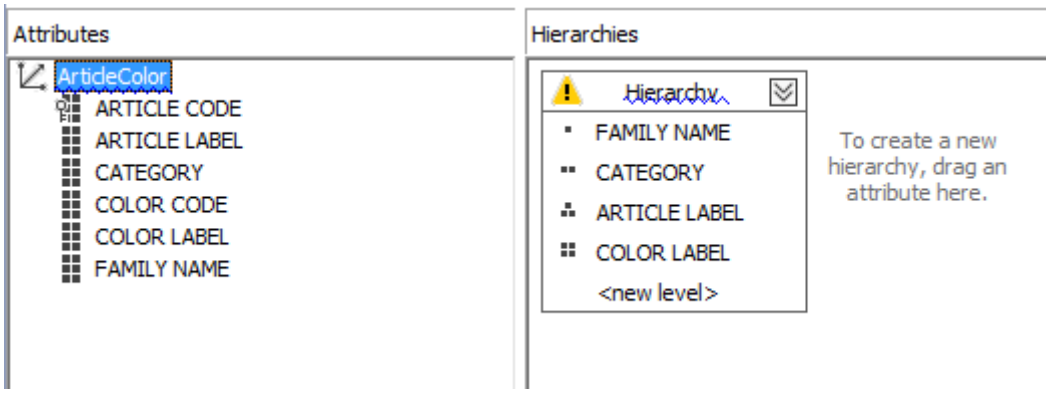


Figure 64 : Attributs et hiérarchie dimension "ArticleColor"

La navigation se présente ainsi :

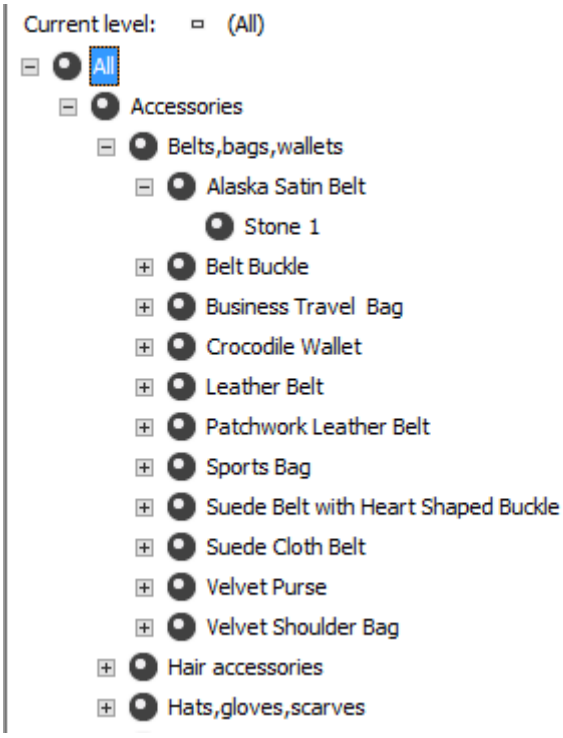


Figure 65 : Navigation dimension "ArticleColor"

4. Cube et mesures

Le cube regroupe en lui un ensemble de mesures correspondant à des valeurs stockées chacune dans leur cellule. Ainsi le cube est associé à un fait, qui correspond au sujet principal d'une analyse d'aide à la décision en d'autres termes, le jeu de mesures manipulées par le cube.

Au sein de notre projet. On trouve la table SHOP_FACTS fournit déjà des mesures comme « Margin », « Quantity Sold » et « Amount Sold », d'où SHOP_FACTS est un fait.

Les pourcentages comme le pourcentage de marge par rapport aux revenus des ventes, le pourcentage de revenu des magasins par rapport aux revenus globaux ou le pourcentage d'article vendu d'une catégorie par rapport au total des quantités vendues sont des mesures précalculées dont on a la possibilité de les ajouter.

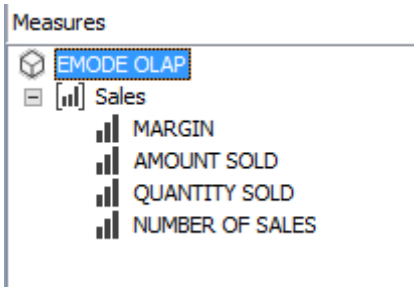


Figure 66 : Mesures du cube

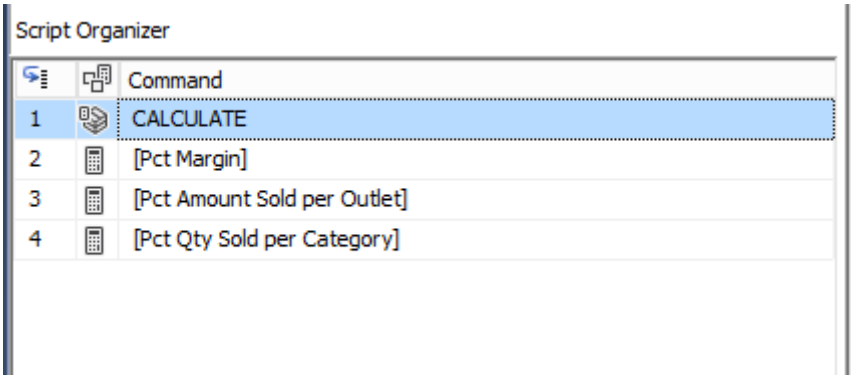


Figure 67 : Mesures précalculées

5. Exploration du cube

Explorer le cube après son déploiement est tout à fait possible à l'aide d'un outil intégré à SSAS.

Voici un exemple de représentation des données :

YEAR	SHOP NAME	AMOUNT SOLD
1999	e-Fashion Austin	561123,4
1999	e-Fashion Boston Newbury	238818,7
1999	e-Fashion Chicago 33rd	737914,200000001
1999	e-Fashion Colorado Springs	448301,500000001
1999	e-Fashion Dallas	427244,699999999
1999	e-Fashion Houston	529078,499999999
1999	e-Fashion Houston Leighton	682230,800000002
1999	e-Fashion Los Angeles	982637,100000003
1999	e-Fashion Miami Sundance	405985,099999999
1999	e-Fashion New York 5th	644635,100000001

Figure 68 : CA par année et par magasin

Partie 3 : La mise en place du reporting

Notre projet consiste à pouvoir exploitée les données par un system d'aide à la décision, et nos 2 parties précédentes se focaliser principalement sur la mise en place d'une base de données optimisée. Ainsi on a mis des ETL afin d'alimenter cette base avec des données qui répond à tous les critères demandés, bien que cette base soit optimisée dans le but de l'exploiter dans les meilleures conditions. Et cette partie finale représente l'exploitation des données pour générer des rapports contenant un certain nombre d'indicateurs. Il existe plusieurs outils pour exploiter ces données.

V. REPORTING SERVICES

Cette partie consiste à la création des rapports avec Reporting Services abrégé SSRS. L'outil fourni par Microsoft avec SQL Server servant à créer et gérer des rapports d'aide à la décision.

On a commencé par créer des projets de type « Analysis Services », qui de vont nous permettre de créer des rapports contenant différents tableaux et graphiques.

a. Rapport 1

Notre premier rapport consiste à afficher le total des revenus des ventes sous des tableau croisé basique.

Le premier est organisé par année et par quarter pour chaque état dans un tableau croisé basique.

Sales By State and Year

STATE / YEAR	1999				2000				2001				TOTAL
	QT 1	QT 2	QT 3	QT 4	QT 1	QT 2	QT 3	QT 4	QT 1	QT 2	QT 3	QT 4	
California	519 220€	441 494€	394 309€	349 188€	650 715€	529 256€	760 442€	842 267€	729 745€	789 398€	775 766€	697 770€	7 479 569€
Colorado	131 797€	129 076€	85 621€	101 807€	189 131€	157 337€	192 267€	229 654€	204 754€	213 663€	232 889€	192 279€	2 060 275€
DC	208 324€	179 863€	131 687€	173 336€	279 490€	263 486€	288 926€	383 257€	279 008€	263 098€	271 645€	239 831€	2 961 950€
Florida	137 530€	121 170€	50 926€	96 359€	174 276€	147 358€	121 314€	218 301€	203 882€	221 469€	215 569€	171 003€	1 879 159€
Illinois	256 454€	241 149€	107 006€	133 306€	334 297€	254 722€	230 573€	331 067€	255 658€	354 724€	273 186€	250 517€	3 022 658€
Massachusetts	92 596€	70 903€	12 066€	63 255€				157 719€	220 301€	220 528€	237 464€	208 877€	1 283 707€
New York	555 983€	479 962€	257 114€	374 637€	683 971€	692 513€	501 220€	885 800€	747 161€	855 617€	914 247€	633 998€	7 582 221€
Texas	758 796€	615 077€	329 113€	496 692€	1 014 293€	795 979€	784 560€	1 138 056€	1 102 481€	1 088 221€	1 032 629€	961 768€	10 117 664€
TOTAL	178 186€	185 423€	115 918€	141 648€	226 268€	220 478€	149 159€	327 991€	288 260€	260 420€	250 330€	298 916€	36 387 203€

Figure 69 : SSRS Rapport 1 Table croisé 1

Le deuxième illustre les ventes par mois et par état avec les années comme section.

Sales by state and date

Year : 1999

STATE / MONTH	January	February	March	April	May	June	July
California	178 186,40€	120 204,50€	220 829,50€	185 423,30€	163 346,30€	92 724,60€	115 918,10€
Colorado	41 686,80€	35 181,30€	54 928,80€	55 378,40€	43 661,00€	30 036,90€	33 139,60€
DC	74 120,30€	48 128,20€	86 075,90€	82 401,20€	60 683,00€	36 778,90€	48 804,30€
Florida	55 731,50€	31 099,40€	50 698,80€	41 044,10€	49 189,70€	30 936,50€	23 097,90€
Illinois	101 985,00€	63 674,10€	90 794,70€	87 755,90€	95 828,90€	57 563,90€	42 213,90€
Massachusetts	38 797,60€	27 822,70€	25 975,20€	24 839,00€	28 984,50€	17 079,20€	10 994,50€
New York	222 473,80€	124 299,80€	209 209,50€	174 392,10€	179 540,80€	126 028,60€	128 788,90€
Texas	290 559,80€	179 663,20€	288 572,70€	244 025,80€	244 380,90€	126 669,90€	122 946,30€
Total	1 003 541,20€	630 073,20€	1 027 085,10€	895 259,80€	865 615,10€	517 818,50€	525 903,50€

Year : 2000

STATE / MONTH	January	February	March	April	May	June	July
California	226 267,50€	128 909,10€	295 538,20€	220 477,70€	182 835,10€	125 943,10€	149 158,50€
Colorado	70 036,30€	41 300,70€	77 794,40€	64 917,70€	54 591,90€	37 827,50€	47 109,10€
DC	95 960,30€	55 960,80€	127 568,40€	99 078,30€	93 787,60€	70 619,90€	61 389,00€
Florida	73 919,00€	31 270,60€	69 086,20€	53 238,20€	59 210,80€	34 909,30€	39 069,50€
Illinois	156 927,30€	57 663,10€	119 706,40€	100 256,90€	96 982,70€	57 482,40€	66 726,20€
Massachusetts							

Figure 70 : SSRS Rapport 1 Table croisé 2

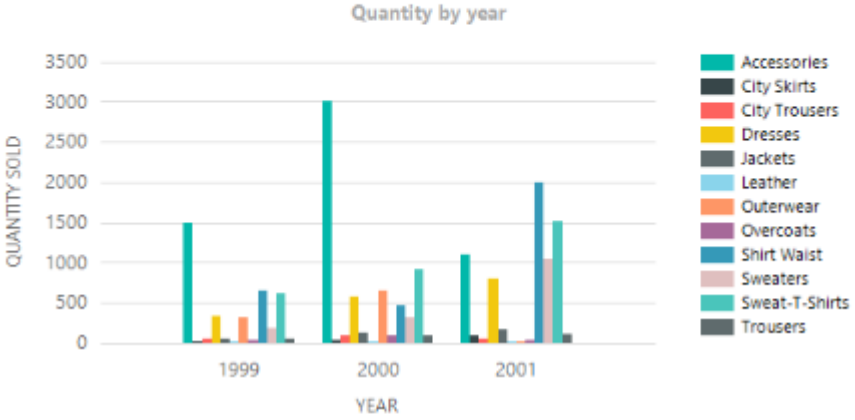
b. Rapport 2

Le deuxième rapport contient deux représentions des données, un tableau simple et le graphe associé :

- Le tableau contient la quantité vendue d'article par type d'article, par catégorie d'article en fonction des différentes années.
- Le graphe associé présente la quantité vendue d'article par catégorie d'article en fonction des années.

Quantity Sold By Category

Store name : e-Fashion Austin



FAMILY NAME	CATEGORY	YEAR	QUANTITY SOLD
Accessories	Belts,bags,wallets	1999	269
		2000	376
		2001	327
		TOTAL BY YEAR	972
	Hair accessories	1999	44
		2000	72
		2001	57
		TOTAL BY YEAR	173
	Hats,gloves,scarves	1999	275
		2000	668
		2001	21

Figure 71 : SSRS Rapport 2 Table et graphe

c. Rapport 3

Le dernier rapport se compose d'un tableau simple et un graphe associé :

- Le tableau simple nous présente la marge et le chiffre d'affaire en fonction des années, boutiques, ville et état.
- Le graphe présente un pourcentage du chiffre d'affaires selon les états

Sales By State and Year

STATE	CITY	STORE NAME	YEAR	SALES RV	MARGIN
California	Los Angeles	e-Fashion Los Angeles	1999	982 637,10€	443 542,00€
			2000	1 581 616,00€	603 636,00€
			2001	1 656 675,70€	618 069,00€
			TOTAL	4 220 928,80€	1 665 247,00€
		TOTAL		4 220 928,80€	1 665 247,00€
	San Francisco	e-Fashion San Francisco	1999	721 573,70€	330 030,00€
			2000	1 201 063,50€	470 752,00€
			2001	1 336 003,30€	500 926,00€
			TOTAL	3 258 640,50€	1 301 708,00€
		TOTAL		3 258 640,50€	1 301 708,00€
	TOTAL			7 479 569,30€	2 966 955,00€
Colorado	Colorado Springs	e-Fashion Colorado Springs	1999	448 301,50€	203 225,00€
			2000	768 389,50€	293 703,00€
			2001	843 584,20€	309 066,00€
			TOTAL	2 060 275,20€	805 994,00€
		TOTAL		2 060 275,20€	805 994,00€
	TOTAL			2 060 275,20€	805 994,00€

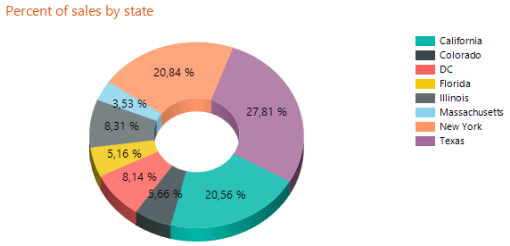


Figure 72 : SSRS Rapport 3 Table et graphe

VI. UNIVERS BUSNIESS OBJECTS

Business Objects est une suite de logiciels d'aide à la décision et comprend des logiciels pour générer des tableaux de bord par exemple. L'objectif est ainsi de permettre à des utilisateurs non informaticiens d'extraire des informations d'une base de données en vue de constituer des rapports.

Avant de pouvoir créer ces rapports avec les outils Business Objects, il est nécessaire de créer un univers. Il s'agit d'une couche intermédiaire qui masque les aspects techniques pour présenter à l'utilisateur une vision métier des données, en utilisant des termes professionnels et non des termes techniques. Généralement, un univers correspond à un domaine fonctionnel précis comme les ressources humaines ou la gestion commerciale.

Chaque univers est composé d'objets regroupés en classes et sous classes. Un objet correspond aux différentes informations de la base de données.

C'est par cette couche que l'utilisateur, via Web Intelligence, pourra consulter les données qu'il désire interroger. Pour la génération des rapports, l'étape de création de l'univers est primordiale.

Pour les besoins du projet, il faut implémenter le modèle en étoile si dessous :

Figure 61 – Modèle en étoile

Le modèle étant implémenté en étoile, nous n'avons pas eu à faire face aux problèmes de boucles.

1. Création de l'univers

L'univers créé s'organise de la manière suivante :

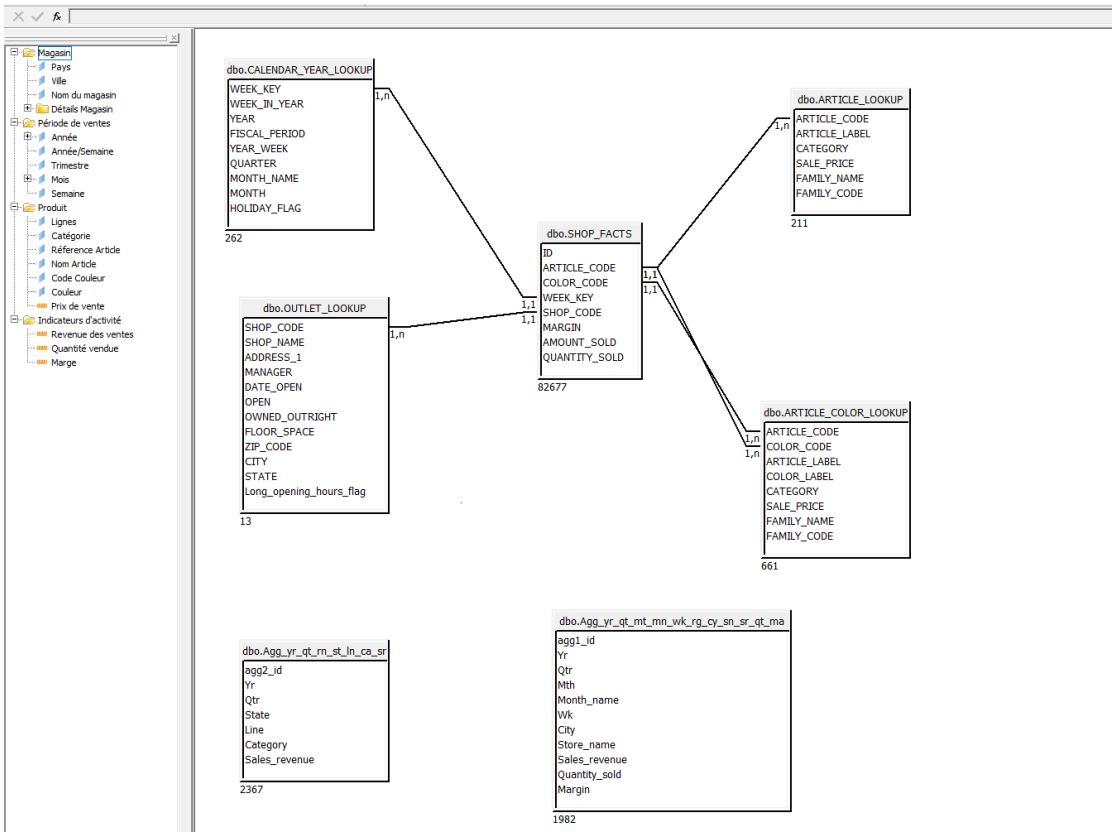


Figure 73 : Univers modèle en étoile

Classe magasin : se base sur les données de la table OUTLET_LOOKUP

Classe Période de Vente : se base sur les données de la table CALENDAR_YEAR_LOOKUP

Classe Produit : se base sur les données des tables ARTICLE_LOOKUP et

ARTICLE_COLOR_LOOKUP

Classe Indicateurs d'Activité : se base sur les données de la table SHOP_FACTS

Ainsi on peut voir le contenu de chaque classe dans la figure ci-dessous :

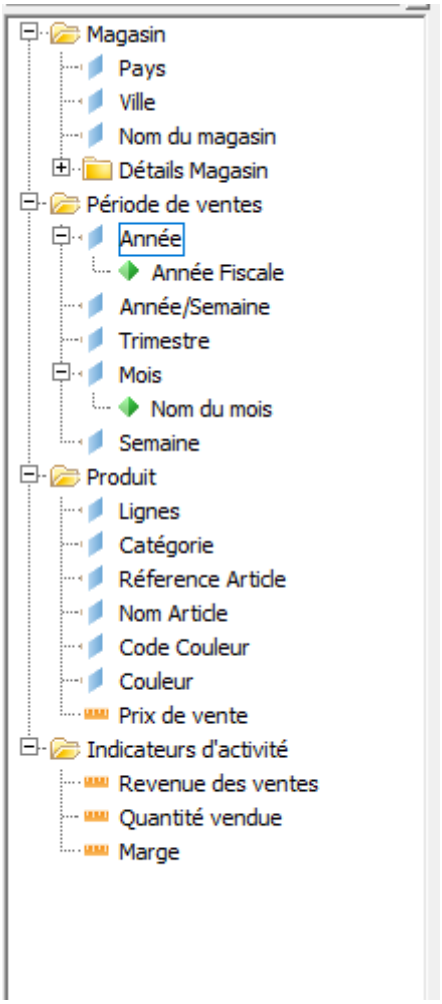


Figure 74 : Structure de l'univers

2. Navigation agrégée

Pour certains objets de l'univers, il est possible de mettre en place la navigation agrégée. Elle va permettre d'optimiser les requêtes SQL. En effet, la table la plus appropriée sera choisie en fonction du choix des objets sélectionné dans l'éditeur de requête.

La navigation agrégée est possible grâce à la fonction :

@Aggregate_Aware(Définition1, Définition2,...) avec les définitions dans l'ordre de la plus agrégée à la moins agrégée.

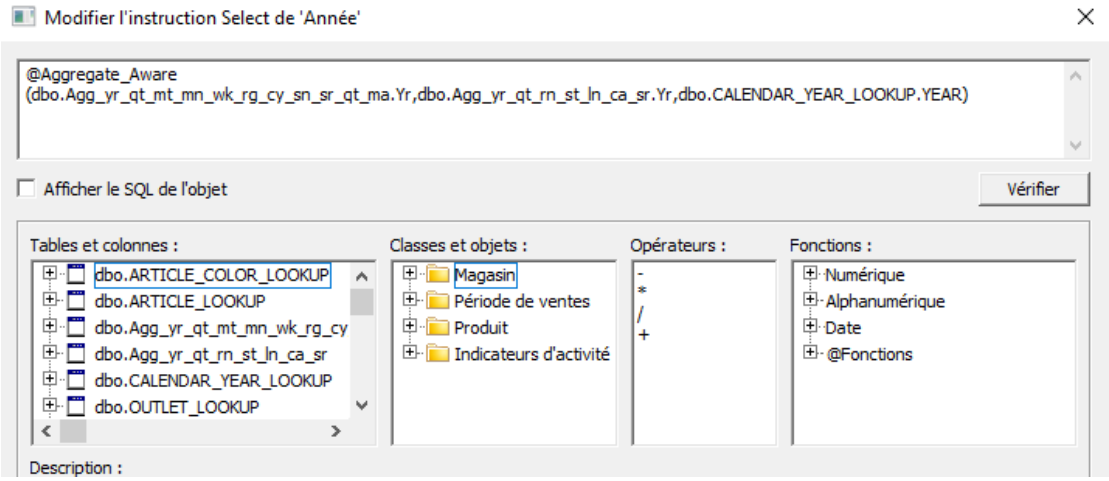


Figure 75 : Exemple d'agrégation

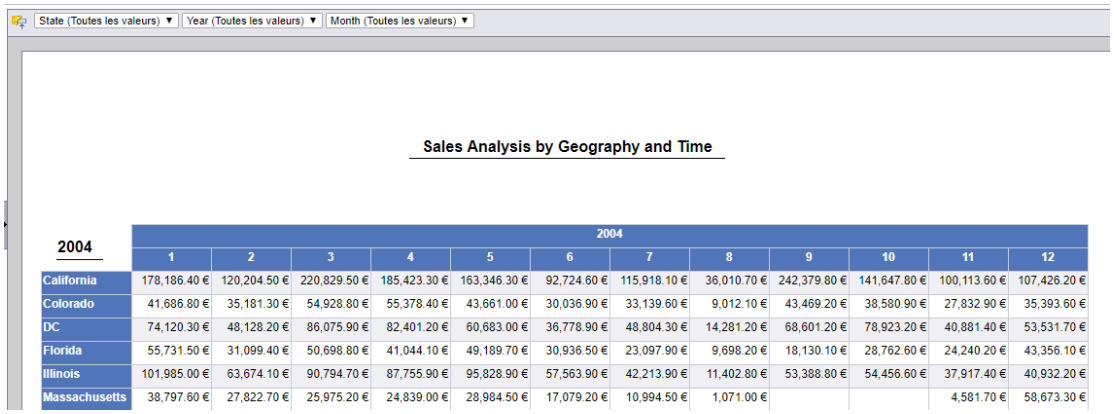
VII. WEB INTELLIGENCE

Dans cette partie nous allons créer des rapports à l'aide de Web Intelligence. Après avoir sélectionné l'univers de notre choix nous pouvons éditer des tableaux de bord à partir du contenu de l'entrepôt de données. Tous les documents sont en mode page et on a changé les modes de portrait a paysage ainsi même la forme de la page de A4 à A3 pour mieux présenter les données.

1. Premier Document :

a. Rapport 1

Le premier rapport doit afficher le revenu des ventes selon le mois, l'année et l'état.



State (Toutes les valeurs) | Year (Toutes les valeurs) | Month (Toutes les valeurs)

Sales Analysis by Geography and Time

2004	2004											
	1	2	3	4	5	6	7	8	9	10	11	12
California	178,186.40 €	120,204.50 €	220,829.50 €	185,423.30 €	163,346.30 €	92,724.60 €	115,918.10 €	36,010.70 €	242,379.80 €	141,647.80 €	100,113.60 €	107,426.20 €
Colorado	41,686.80 €	35,181.30 €	54,928.80 €	55,378.40 €	43,661.00 €	30,036.90 €	33,139.60 €	9,012.10 €	43,469.20 €	38,580.90 €	27,832.90 €	35,393.60 €
DC	74,120.30 €	48,128.20 €	86,075.90 €	82,401.20 €	60,683.00 €	36,778.90 €	48,804.30 €	14,281.20 €	68,601.20 €	78,923.20 €	40,881.40 €	53,531.70 €
Florida	55,731.50 €	31,099.40 €	50,698.80 €	41,044.10 €	49,189.70 €	30,936.50 €	23,097.90 €	9,698.20 €	18,130.10 €	28,762.60 €	24,240.20 €	43,356.10 €
Illinois	101,985.00 €	63,674.10 €	90,794.70 €	87,755.90 €	95,828.90 €	57,563.90 €	42,213.90 €	11,402.80 €	53,388.80 €	54,456.60 €	37,917.40 €	40,932.20 €
Massachusetts	38,797.60 €	27,822.70 €	25,975.20 €	24,839.00 €	28,984.50 €	17,079.20 €	10,994.50 €	1,071.00 €			4,581.70 €	58,673.30 €

Figure 76 : WI Doc 1 Rapport 1

b. Rapport 2

Le deuxième rapport affiche les mêmes informations que le premier mais nous y ajoutons un total par magasin et par mois ainsi qu'un diagramme bâtons représentant ces données.

Sales Analysis by Geography and Time

2004	1	2	3	4	5	6	7	8	9	10	11	12	Sum:
California	178,186.40 €	120,204.50 €	220,829.50 €	185,423.30 €	163,346.30 €	92,724.60 €	115,918.10 €	36,010.70 €	242,379.80 €	141,647.80 €	100,113.60 €	107,426.20 €	1,704,210.80 €
Colorado	41,686.80 €	35,181.30 €	54,928.80 €	55,378.40 €	43,661.00 €	30,036.90 €	33,139.60 €	9,012.10 €	43,469.20 €	38,580.90 €	27,832.90 €	35,393.60 €	448,301.50 €
DC	74,120.30 €	48,128.20 €	86,075.90 €	82,401.20 €	60,683.00 €	36,778.90 €	48,804.30 €	14,281.20 €	68,601.20 €	78,923.20 €	40,881.40 €	53,531.70 €	693,210.50 €
Florida	55,731.50 €	31,099.40 €	50,698.80 €	41,044.10 €	49,189.70 €	30,936.50 €	23,097.90 €	9,698.20 €	18,130.10 €	28,762.60 €	24,240.20 €	43,356.10 €	405,985.10 €
Illinois	101,985.00 €	63,674.10 €	90,794.70 €	87,755.90 €	95,828.90 €	57,563.90 €	42,213.90 €	11,402.80 €	53,388.80 €	54,456.60 €	37,917.40 €	40,932.20 €	737,914.20 €
Massachusetts	38,797.60 €	27,822.70 €	25,975.20 €	24,839.00 €	28,984.50 €	17,079.20 €	10,994.50 €	1,071.00 €			4,581.70 €	58,673.30 €	238,818.70 €
New York	222,473.80 €	124,299.80 €	209,209.50 €	174,392.10 €	179,540.80 €	126,028.60 €	128,788.90 €	39,452.60 €	88,872.40 €	138,460.70 €	118,595.40 €	117,581.20 €	1,667,695.80 €
Texas	290,559.80 €	179,663.20 €	288,572.70 €	244,025.80 €	244,380.90 €	126,669.90 €	122,946.30 €	52,827.80 €	153,339.30 €	174,374.60 €	129,861.60 €	192,455.50 €	2,199,677.40 €
Sum :	1,003,541.20 €	630,073.20 €	1,027,085.10 €	895,259.80 €	865,615.10 €	517,818.50 €	525,903.50 €	173,756.40 €	668,180.80 €	655,206.40 €	484,024.20 €	649,349.80 €	8,095,814.00€

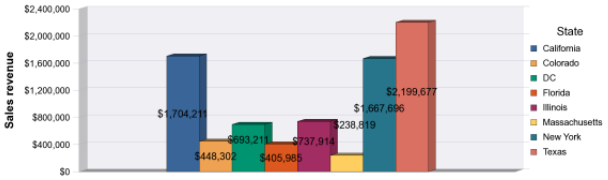


Figure 77 : WI Doc 1 Rapport 2

c. Rapport 3

Le troisième rapport est simplement une page de garde avec des liens qui nous mène vers les rapports qui sont disponibles dans le document et se trouve à la première place.

Analys multiples des ventes

1. Analyse mensuelle des ventes par tableau croisé

2. Analyse mensuelle des ventes par graphique

Figure 78 : WI Doc 1 Page de garde

2. Second Document :

a. Rapport 1

Le premier rapport affiche les quantités vendues par année et par état.

Quantity sold by year and state

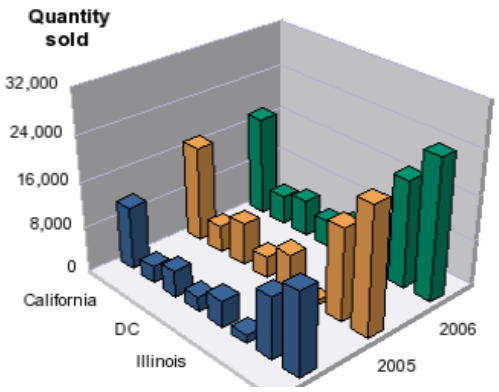
Year	State	Quantity sold
2004	California	11,304
	Colorado	2,971
	DC	4,681
	Florida	2,585
	Illinois	4,713
	Massachusetts	1,505
	New York	10,802
	Texas	14,517
2004	S/Total :	53,078

Figure 79 : WI Doc 2 Rapport 1

b. Rapport 2

Le deuxième rapport affiche les quantités vendues par année et par état à l'aide d'un histogramme 3D.

Quantity sold by year and state



	Quantity sold
California	46,074
Colorado	12,787
DC	18,744
Florida	11,267
Illinois	17,976
Massachusetts	7,676
New York	46,358
Texas	62,347

	Quantity sold
2004	53,078

Figure 80 : WI Doc 2 Rapport 2

c. Rapport 3

Le troisième rapport affiche les quantités d'articles vendus par état ainsi que le pourcentage correspondant par rapport au totale des articles vendus, sous forme de tableau simple ainsi qu'un camembert.

Table with percentage and Pie Chart

State	Quantity sold	Pourcentage
California	46,074	20.64%
Colorado	12,787	5.73%
DC	18,744	8.40%
Florida	11,267	5.05%
Illinois	17,976	8.05%
Massachusetts	7,676	3.44%
New York	46,358	20.77%
Texas	62,347	27.93%
	Pourcentage :	100.00%

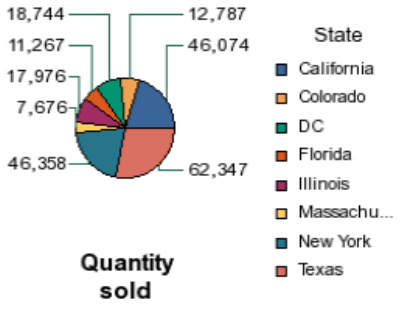


Figure 81 : WI Doc 2 Rapport 3

3. Ajout des contrôles de saisie

Pour chaque rapport, nous avons mis des contrôles de saisie, ainsi on permettra aux utilisateurs de filtrer les données et d'afficher seulement ceux qui correspond a ces exigences. Cela peut être caractérisé par une liste déroulante avec des valeurs, une liste à choix multiples ou un champ de saisie par exemple.

Month (Toutes les valeurs) ▼ State (Toutes les valeurs) ▼ Year (Toutes les valeurs) ▼

2004

	1	2	3	4	5	6	7	8
California	178,186.40 €	120,204.50 €	220,829.50 €	185,423.30 €	163,346.30 €	92,724.60 €	115,918.10 €	36,010.70 €
Colorado	41,686.80 €	35,181.30 €	54,928.80 €	55,378.40 €	43,661.00 €	30,036.90 €	33,139.60 €	9,012.10 €
DC	74,120.30 €	48,128.20 €	86,075.90 €	82,401.20 €	60,683.00 €	36,778.90 €	48,804.30 €	14,281.20 €
Florida	55,731.50 €	31,099.40 €	50,698.80 €	41,044.10 €	49,189.70 €	30,936.50 €	23,097.90 €	9,698.20 €
Illinois	101,985.00 €	63,674.10 €	90,794.70 €	87,755.90 €	95,828.90 €	57,563.90 €	42,213.90 €	11,402.80 €
Massachusetts	38,797.60 €	27,822.70 €	25,975.20 €	24,839.00 €	28,984.50 €	17,079.20 €	10,994.50 €	1,071.00 €
New York	222,473.80 €	124,299.80 €	209,209.50 €	174,392.10 €	179,540.80 €	126,028.60 €	128,788.90 €	39,452.60 €
Texas	290,559.80 €	179,663.20 €	288,572.70 €	244,025.80 €	244,380.90 €	126,669.90 €	122,946.30 €	52,827.80 €
Sum :	1,003,541.20 €	630,073.20 €	1,027,085.10 €	895,259.80 €	865,615.10 €	517,818.50 €	525,903.50 €	173,756.40 €

Figure 82 : WI Doc 1 Rapport 1 avec filtre

State (Toutes les valeurs) ▼ Year (Toutes les valeurs) ▼ Month (Toutes les valeurs) ▼

Sales Analysis by Geography and Time

2004	2004											
	1	2	3	4	5	6	7	8	9	10	11	12
California	178,186.40 €	120,204.50 €	220,829.50 €	185,423.30 €	163,346.30 €	92,724.60 €	115,918.10 €	36,010.70 €	242,379.80 €	141,647.80 €	100,113.60 €	107,426.20 €
Colorado	41,686.80 €	35,181.30 €	54,928.80 €	55,378.40 €	43,661.00 €	30,036.90 €	33,139.60 €	9,012.10 €	43,469.20 €	38,580.90 €	27,832.90 €	35,393.60 €
DC	74,120.30 €	48,128.20 €	86,075.90 €	82,401.20 €	60,683.00 €	36,778.90 €	48,804.30 €	14,281.20 €	68,601.20 €	78,923.20 €	40,881.40 €	53,531.70 €
Florida	55,731.50 €	31,099.40 €	50,698.80 €	41,044.10 €	49,189.70 €	30,936.50 €	23,097.90 €	9,698.20 €	18,130.10 €	28,762.60 €	24,240.20 €	43,356.10 €
Illinois	101,985.00 €	63,674.10 €	90,794.70 €	87,755.90 €	95,828.90 €	57,563.90 €	42,213.90 €	11,402.80 €	53,388.80 €	54,456.60 €	37,917.40 €	40,932.20 €
Massachusetts	38,797.60 €	27,822.70 €	25,975.20 €	24,839.00 €	28,984.50 €	17,079.20 €	10,994.50 €	1,071.00 €			4,581.70 €	58,673.30 €

Figure 83 : WI Doc 1 Rapport 2 avec filtre

VIII. EXCEL

Excel 2013 offre diverses fonctionnalités dont une dédiée à l'aide à la décision. Il s'agit de « Power Pivot ». Ce module permet d'importer des lignes de données dans un classeur à partir de sources de données et de les utiliser pour créer des rapports à l'aide de tableaux croisés ou de graphiques.

Il est possible d'utiliser plusieurs connexions pour alimenter ces rapports sous Excel 2013. En effet, on peut utiliser une connexion directe à une base de données relationnelle ou alors utiliser un cube OLAP. Les limitations matérielles et logicielles nous imposent d'utiliser une connexion directe à une base de données sur un serveur de l'UTBM.

Le modèle de données sur EXCEL est sous forme d'étoile :

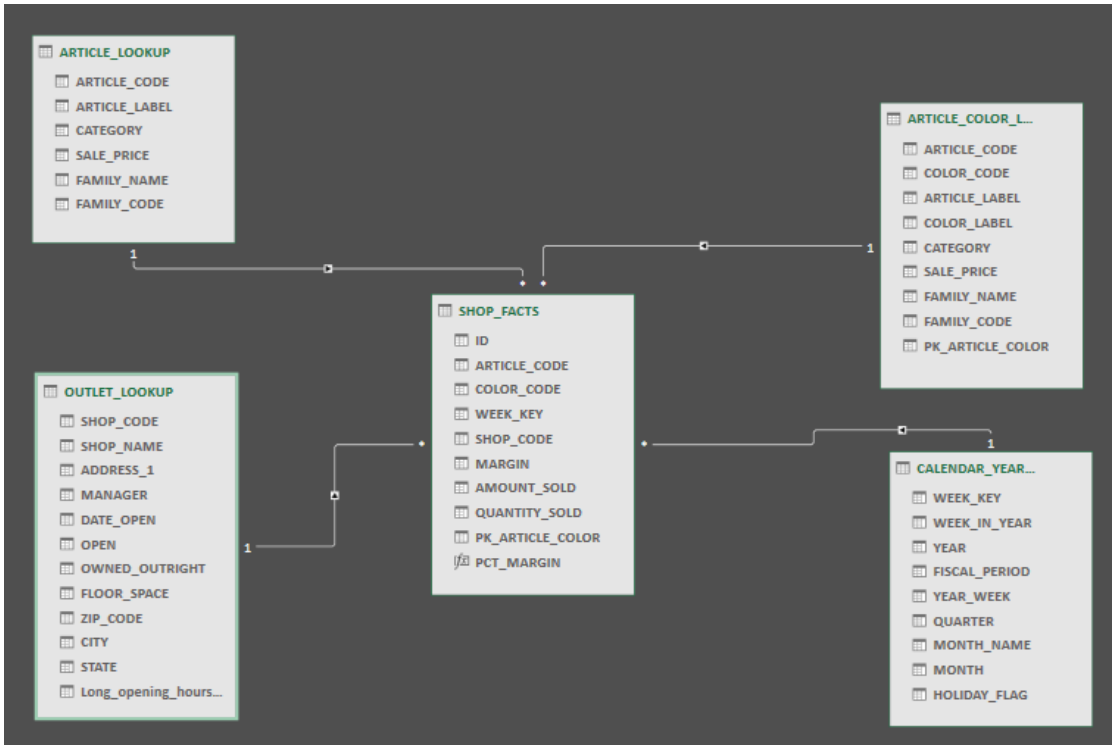
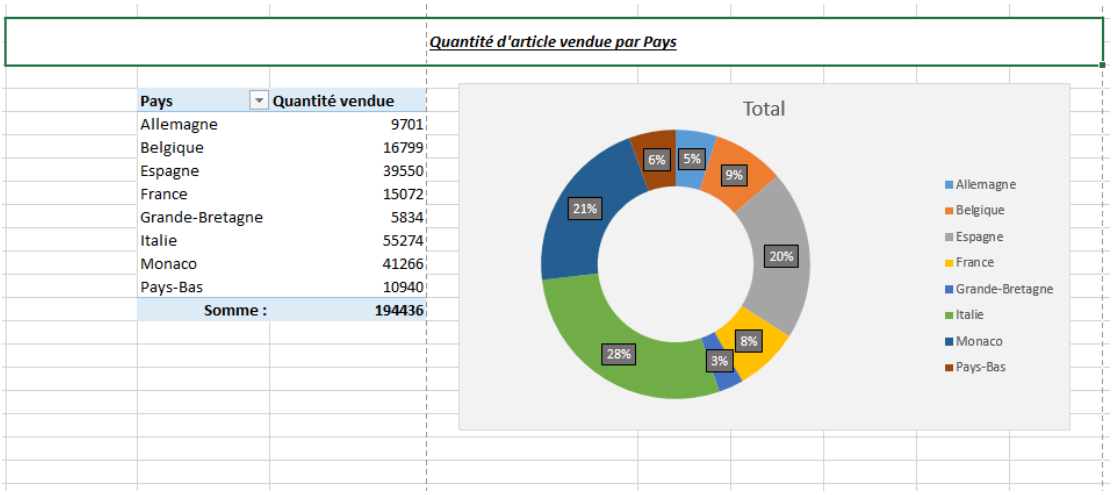


Figure 84 : Modèle de données sous EXCEL

1. Rapports

Les exemples de rapport ci-dessous permettent de montrer les différentes fonctionnalités proposées pour réaliser des rapports.

a. Tableau simple et camembert



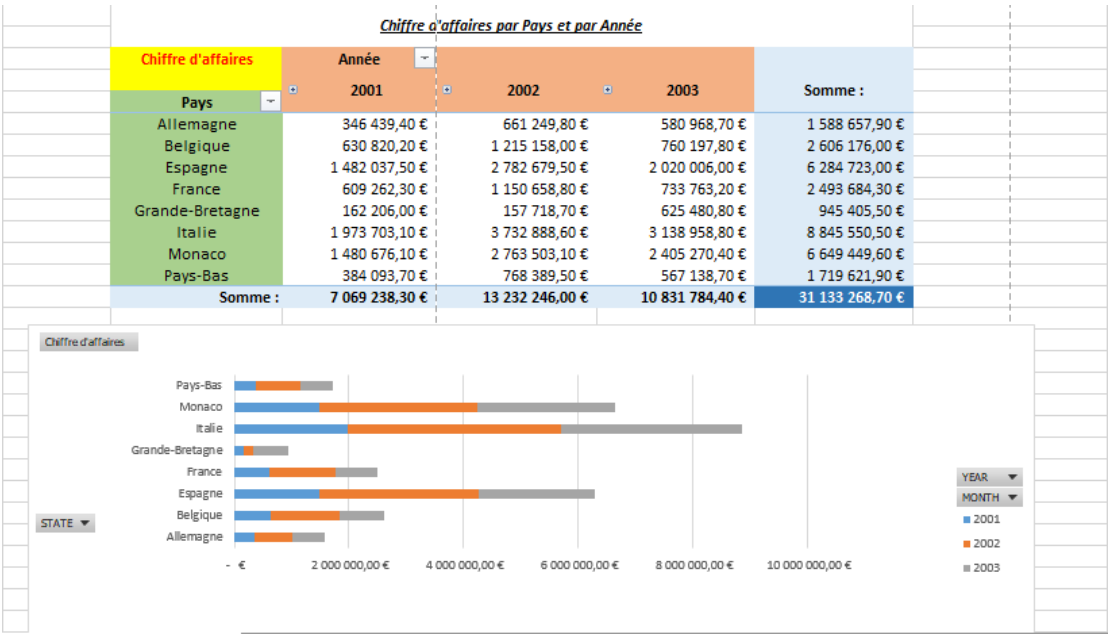
Quantité d'article vendue par Pays

Pays	Quantité vendue
Allemagne	9701
Belgique	16799
Espagne	39550
France	15072
Grande-Bretagne	5834
Italie	55274
Monaco	41266
Pays-Bas	10940
Somme :	194436

Figure 85 : EXCEL Table simple et camembert

Simple tableau qui représente la quantité vendue par pays ainsi qu'un camembert illustrant le pourcentage des ventes par pays.

b. Tableau croisé et graphe



Chiffre d'affaires par Pays et par Année

Chiffre d'affaires / Pays	Année 2001	2002	2003	Somme :
Allemagne	346 439,40 €	661 249,80 €	580 968,70 €	1 588 657,90 €
Belgique	630 820,20 €	1 215 158,00 €	760 197,80 €	2 606 176,00 €
Espagne	1 482 037,50 €	2 782 679,50 €	2 020 006,00 €	6 284 723,00 €
France	609 262,30 €	1 150 658,80 €	733 763,20 €	2 493 684,30 €
Grande-Bretagne	162 206,00 €	157 718,70 €	625 480,80 €	945 405,50 €
Italie	1 973 703,10 €	3 732 888,60 €	3 138 958,80 €	8 845 550,50 €
Monaco	1 480 676,10 €	2 763 503,10 €	2 405 270,40 €	6 649 449,60 €
Pays-Bas	384 093,70 €	768 389,50 €	567 138,70 €	1 719 621,90 €
Somme :	7 069 238,30 €	13 232 246,00 €	10 831 784,40 €	31 133 268,70 €

Figure 86 : EXCEL Table croisé et graphe

Représentation des chiffres d'affaire par année et par pays.

c. Table d'indicateurs

Chiffre d'affaires par magasin	
Magasin	**Chiffre d'affaires**
e-Mode Amsterdam	1 719 621,90 €
e-Mode Berlin	1 588 657,90 €
e-Mode Bruxelles	2 606 176,00 €
e-Mode Florence	2 422 695,30 €
e-Mode Grenade	2 729 079,20 €
e-Mode Londres	945 405,50 €
e-Mode Madrid	3 555 643,80 €
e-Mode Milan	1 686 091,40 €
e-Mode Monaco Golf	2 541 195,00 €
e-Mode Monaco Montecarlo	4 108 254,60 €
e-Mode Paris	2 493 684,30 €
e-Mode Rome Bellavista	2 032 202,80 €
e-Mode Rome Termini	2 704 561,00 €
Total général	**31 133 268,70 €**

Figure 87 : EXCEL Table d'indicateurs

Ce tableau d'indicateurs représente le chiffre d'affaires par magasin en respectant les règles suivantes :

Afficher chaque icône en fonction de ces règles :

Icône			Valeur
	si la valeur est	>=	3000000
	si < 3000000 et	>=	2000000
	si < 2000000		

Figure 88 : EXCEL Règles de la table d'indicateurs

d. Tableau de bord

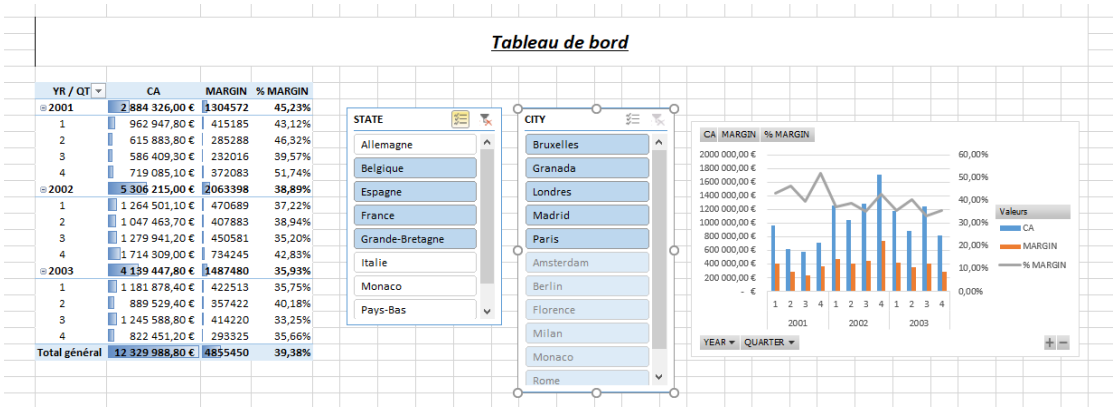


Figure 89 : EXCEL Tableau de bord

Le tableau de bord illustre l'ensemble des informations dont on dispose, ainsi on peut faire nos sélections voulu pour filtrer les données selon le besoin.

2. Power pivot

A l'aide de POWER PIVOT, on a pu monter 3 Rapports avec des fonctionnalité différentes.

- Le premier présente les ventes par payes sous forme de tableau simple, camembert et diagramme bâtons.

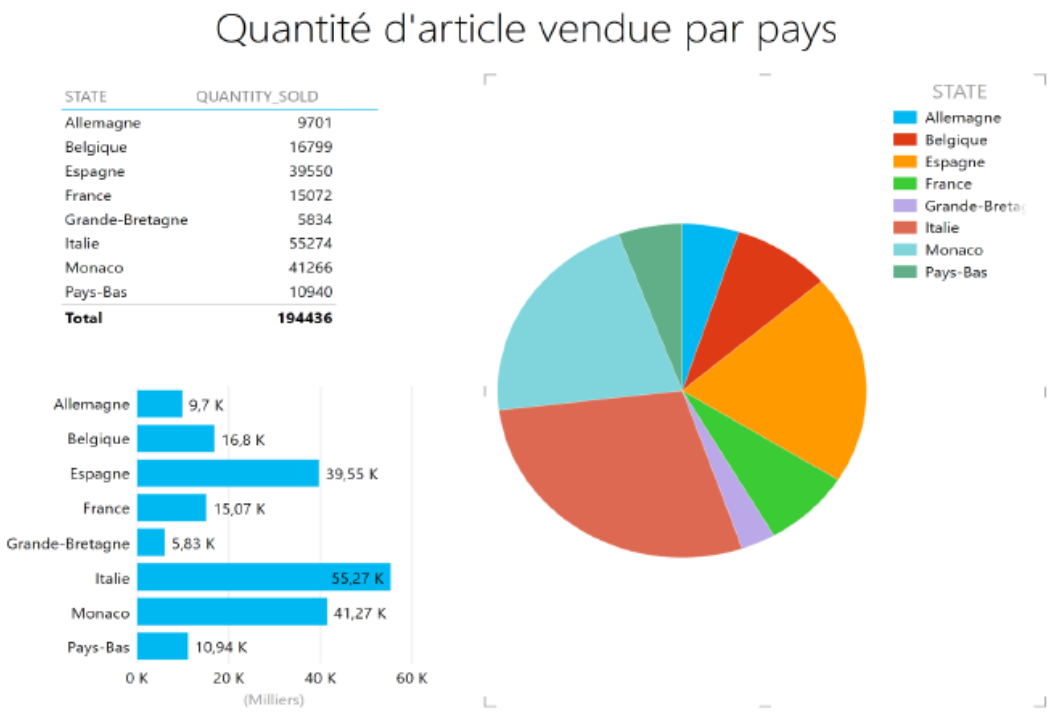


Figure 90 : POWER PIVOT ventes par pays 1

Ainsi on a la possibilité de selectioné le pays qui nous interesse et on aura toutes les representatio filtré.

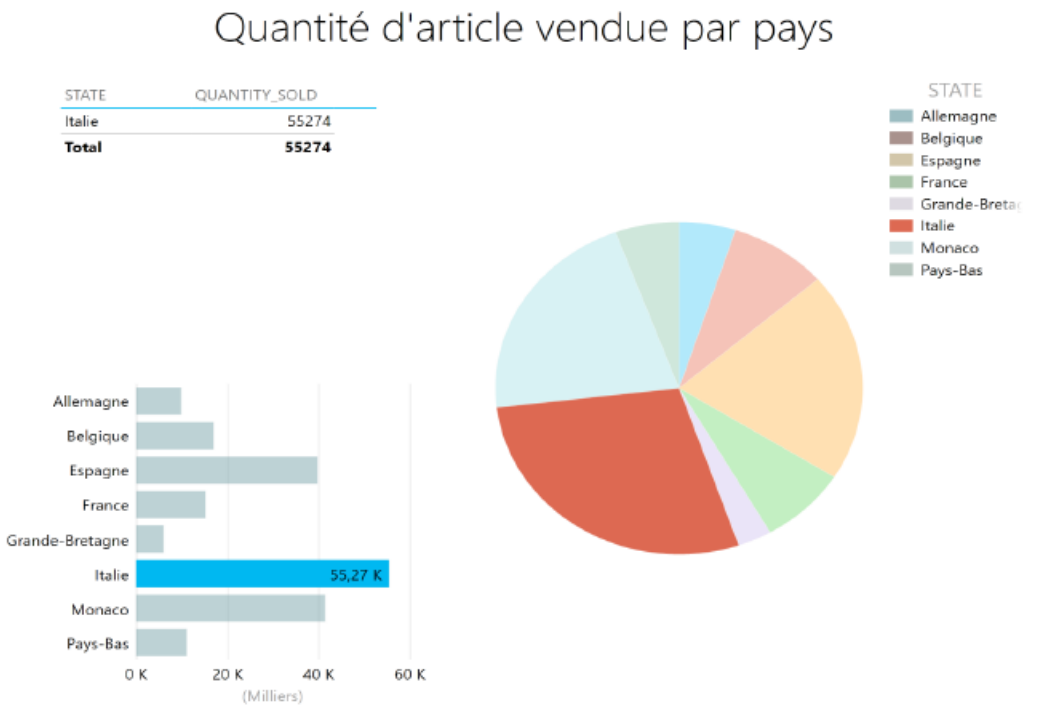


Figure 91 : POWER PIVOT ventes par pays 2

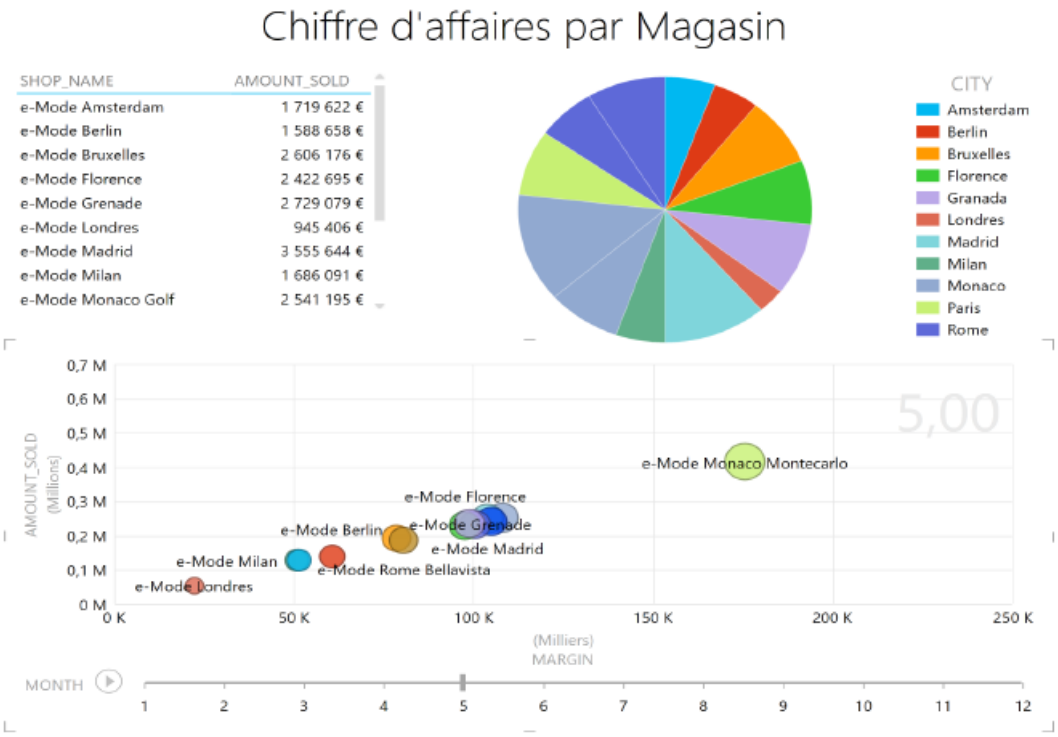


Figure 92 : POWER PIVOT Rapport 2

Pour ce rapport, on trouve les ventes par magasin sur le tableau, ainsi sur le camembert on trouve les ventes par pays et si on y click on accède aux ventes par magasin puis par catégories et par type de produit. Ainsi sur le dernier graphe on peut visualiser l'évolution des ventes par pays dans la dernière année par mois.

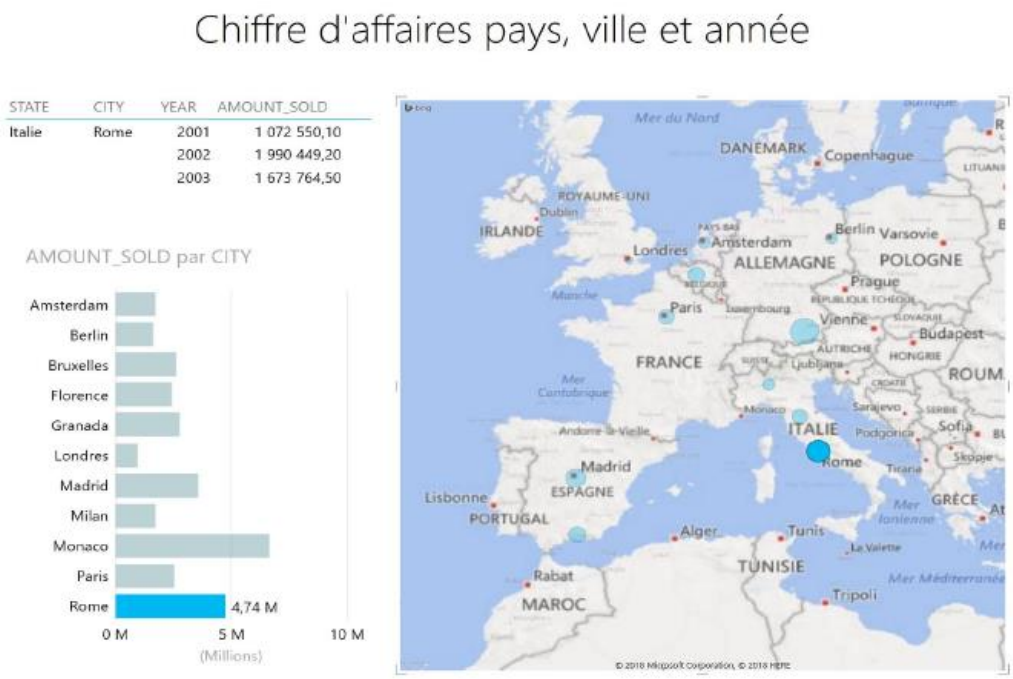


Figure 93 : POWER PIVOT Rapport 3

Pour ce rapport, on a le chiffre d'affaire par pays, ville et année ainsi on peut faire un filtre en sélectionnant la ville voulu sur la carte ou sur le diagramme de bâtons.

IX. QLIKVIEW

Qlikview est un outil d'aide à la décision dans la même lignée de Web Intelligence ou Power Pivot fourni avec Excel 2013. Qliview a la possibilité d'agréger des données issues de sources différentes.

1. Chargement des données

Avant le chargement des données, une connexion à une base de données est nécessaire. Dans le cadre du projet, nous utilisons une connexion à une base de données sur un serveur de l'UTBM. Cette connexion se caractérise par une instruction avec différents paramètres mais un assistant permet de définir ces paramètres.

Avec Qlikview le chargement des données se fait à l'aide de script.

Main | SHOP_FACTS | CALENDAR_YEAR_LOOKUP | ARTICLE_COLOR_LOOKUP | OUTLET_LOOKUP | ARTICLE_LOOKUP

```
1  QUALIFY |
2      ARTICLE_LABEL,
3      CATEGORY,
4      COLOR_LABEL,
5      FAMILY_CODE,
6      FAMILY_NAME,
7      SALE_PRICE;
8  LOAD ("ARTICLE_CODE" & '-' &"COLOR_CODE") as %ArticleColorID,
9      "ARTICLE_LABEL",
10     CATEGORY,
11     "COLOR_LABEL",
12     "FAMILY_CODE",
13     "FAMILY_NAME",
14     "SALE_PRICE";
15 SQL SELECT "ARTICLE_CODE",
16     "ARTICLE_LABEL",
17     CATEGORY,
```

Figure 94 : Qlikview script ACL

Une fois toutes les tables importées, nous obtenons le modèle en étoile souhaité.

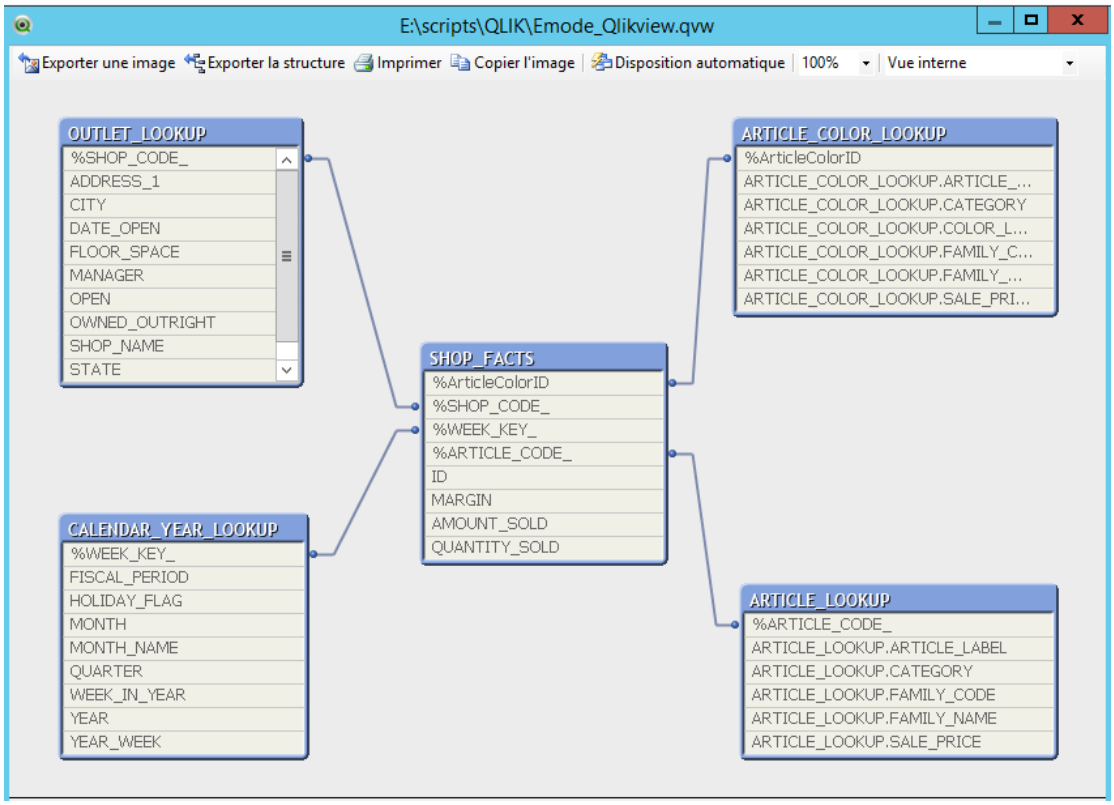


Figure 95 : Qlikview modèle

2. Tableau de bord

Nous pouvons dès à présent créer notre tableau de bord. Il est constitué de six tables de sélection, une zone de recherche rapide et une zone de sélection active afin de visualiser les filtres du rapport et enfin des tableaux simples, croisés et graphiques sont créés.

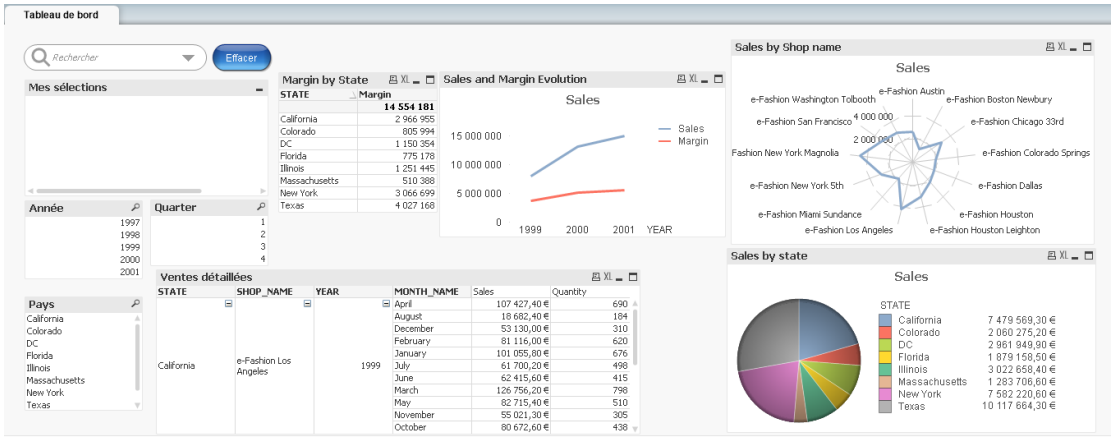


Figure 96 : Qlikview tableau de bord

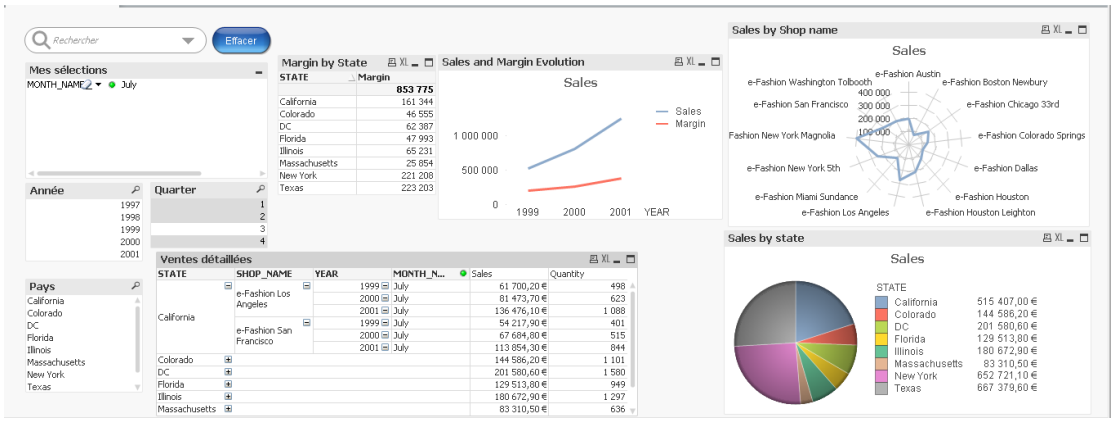


Figure 97 : Qlikview tableau de bord avec filtre par mois

Conclusion

Notre projet au sein de l'Unité de Valeur BD51 est basé sur l'informatique décisionnelle « Business Intelligence ». Lors de la réalisation de ce projet, on a expérimenté les notions reprise en cours et mis en fonction les compétences acquises en Travaux pratiques afin de réaliser un système d'aide à la décision. Notre travail commence par le transfert des données à un nouvel environnement, d'assurer les mises à jour et la cohérence des données, puis agréger et optimiser ces données et finir par les exploités en pleine productivité.

Lors de la réalisation de ce projet, on a pu rencontrer des difficultés qui ont fait appelle a plus de patience et concentration afin de les surmonter, surtout au niveau de la partie ETL. Ainsi cette partie est la base pour tous notre travail, ou il fallait assurer la cohérence des données en imaginant tous les scénarii possibles et de les traiter pour assure à la fin que les rapports seront corrects et productifs.

Également, on a pu se familiariser et découvrir plus les outils et de les manipuler longuement pour avoir une idée sur le rendement de ces derniers même si nous n'avons pas envisagé toutes les fonctionnalités possibles. Mais ceci nous a permis d'avoir une vision globale d'un panel d'outils et d'être rapidement opérationnel sur ces applications qui peuvent être un point stratégique d'une entreprise.

Finalement, la partie exploitation des données qui consiste à la création de rapports et de tableaux de bords pertinents qu'on l'a réalisé avec différents outils avec le principal bute représentation compréhensible, claire et pertinente des données pour des utilisateurs n'ayant pas de connaissances techniques.